TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Niko Lehto

Lohkoketjuavaimen varmistettu ja suojattu hallintaympäristö

Diplomityö
Tietotekniikan tutkinto-ohjelma
Toukokuu 2019

Lehto N. (2019) Lohkoketjuavaimen varmistettu ja suojattu hallintaympäristö. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 47 s.

TIIVISTELMÄ

Lohkoketjuteknologia mahdollistaa käyttäjien välisiä transaktioita, sopimuksia ja automaattisesti suoritettavia ohjelmia ilman keskitettyä palvelinta tai kolmatta osapuolta. Samanaikaisesti kun lohkoketjuteknologia vapauttaa sitoutumasta kolmansiin osapuoliin, avainten hallinnasta on muodostumassa merkittävä haaste järjestelmän loppukäyttäjille. Keskitetyissä järjestelmissä käyttäjien avaintietueet ovat yleisesti kolmannen osapuolen hallinnassa ja palautettavissa. Lohkoketjuteknologian ansiosta loppukäyttäjien on mahdollista säilyttää itsellään identifiointiin käytettäviä avaintietueita. Samalla kun kolmansien osapuolten mahdollisuudet väärinkäytöksille vähentyvät, kukaan muu kuin käyttäjä itse ei ole enää vastuussa avaimen turvallisesta säilyttämisestä. Loppukäyttäjän vastuulle jää, että avaimesta on olemassa varmuuskopio, avaimen muodostamiseen käytetään kryptografisesti turvallista satunnaislukugeneraattoria ja ettei kenelläkään toisella ole mahdollisuutta käyttää avainta. Työssä esitellään malli ja ohjelma, jonka avulla avainta säilytetään luotetussa suoritusympäristössä siten että ainoastaan suorittavalla ohjelmalla on mahdollisuus käsitellä avainta. Lisäksi menetelmä tarjoaa mahdollisuuden varmuuskopioida avaimen toisistaan riippumattomia osia ulkopuolisiin tietueisiin, esimerkiksi pilvipalveluihin. Työn toteutus hyödyntää Intel SGX suoritusympäristöä suojaamaan Ethereumissa käytettävää yksityisavainta.

Avainsanat: vertaisvaluutta, lompakkosovellus, kryptografia, Ethereum, Intel SGX

Lehto N. (2019) Secured and protected management environment for blockchain keys. University of Oulu, Degree Programme in Computer Science and Engineering, 47 p.

ABSTRACT

Blockchain technology enables peer-to-peer transactions, contracts and automatically executable programs without centralized server or third party. While blockchain technology removes need of trust to third party, key management is becoming a major challenge for end-users of system. In centralized systems, users' key records are generally managed and recoverable by third party. Blockchain technology enables end-users to retain key-records used for identification. While opportunities of third party for abuse is diminished, anyone else than the user himself is no longer responsible for the safe storage of the key record. It is the responsibility of end-user to have a backup of the key, use of a cryptographically secure random number generator to generate the key and ensure that no one else has the ability to use that key. The work introduces a model and a program that stores the key in a trusted execution environment so that only the executing program has privilege to utilize the key. In addition, the method provides the ability to back up key as an independent parts to external records, such as cloud services. Implementation of the work utilizes Intel SGX environment to protect Ethereum private key.

Keywords: peer-to-peer electronic cash, crypto wallet, cryptography, Ethereum, Intel SGX

SISÄLLYSLUETTELO

ALKULAUSE

Tämä diplomityö on tehty VTT:n Cryptovault projektissa. Haluan kiittää kaikkia ihmisiä, jotka ovat tukeneet ja opastaneet diplomityötäni. Haluan välittää kiitokset VTT:lle miellyttävästä ympäristöstä tehdä diplomityötä. Erityisesti haluan osoittaa kiitokset Kimmo Haluselle ja Markku Kylänpäälle mielenkiintoisesta diplomityön aiheesta, Kimmolle, Markulle ja Sami Lehtoselle johdatuksesta aiheeseen sekä projektin valmistumisesta. Lopuksi haluan kiittää Kimmoa ja Juha Partalaa diplomityön ohjaamisesta.

Oulu, 28. toukokuuta 2019

Niko Lehto

LYHENTEIDEN JA MERKKIEN SELITYKSET

BIP	Bitcoin Improvement Proposal, Bitcoin-yhteisön dokumenttiformaatti parannusehdotusten esittelemiselle
DApp	Decentralized Application, keskittämätön sovellus
DSA	Digital Signature Algorithm, digitaalinen allekirjoitus algoritmi
ECC	Elliptic Curve Cryptography, elliptisen käyrän kryptografia
ECDLP	Elliptic Curve Discrete Logarithm Problem, diskreetin logaritmin ongelma elliptisellä käyrällä
ECDSA	Elliptic Curve Digital Signature Algorithm, elliptisen käyrän allekirjoitusalgoritmi
EDL	Enclave Definition Language, erillisohjelman kuvauskieli
EOA	Externally Owned Accounts, Ethereumin käyttäjän omistama tavallinen tili
EVM	Ethereum Virtual Machine, Ethereum-virtuaalikone
Intel SGX	Intel Software Guard Extensions, Intel-suorittimen suojattu käskykanta
IoT	Internet of Things, esineiden internet
KDF	Key Derivation function, avaimen muodostus funktio
OTP	One-time pad, kertakäyttöisiin avaimiin perustuva todistettavasti turvallinen salausmenetelmä
Pk	Public key, julkinen avain
PKDF2	Password-Based Key Derivation Function 2, standardi salasanaan perustuvalle avaimen muodostus funktiolle
PoA	Proof of Authority, valtuuden todistus
PoS	Proof of Stake, osuuden todistus
PoW	Proof of Work, työn todistus
RLP	Recursive Length Prefix, rekursiivinen pituuden etuliite
RSA	Rivest-Shamir-Adleman, julkisen avaimen salausalgoritmi
SHA	Secure Hash Function, kryptografinen hash-funktio
Sk	Secret key, salainen avain
SSSS	Shamir Secret Sharing Scheme, Shamirin salaisuuden jako menetelmä
TPM	Trusted Platform Module, turvapiiri
$\mathbb{N}_{>0}$	Positiivisten luonnollisten lukujen joukko $\{1, 2, 3, \ldots\}$
$\mathbb{P}$	Alkulukujen joukko $\{2, 3, 5, 7, \ldots\}$
$\{0,1\}^n$	n-mittaisten binäärivektoreiden joukko
$f()$	Funktio f
$f^{-1}()$	Funktion f käänteisfunktio
$[a, b]$	Suljettu väli elementistä a elementtiin b, joukko käsittää elementit $\{a, \ldots, b\}$
$a \| b$	Elementtien a ja b yhteenketjutus
$a \mid a \leq b$	a, jossa a on pienempi tai yhtä suuri kuin b

$a \in \mathbb{N}_{>0}$	a kuuluu joukkoon $\mathbb{N}_{>0}$
$\mathbf{X}$	Vektori $\mathbf{X}$
$\lvert\mathbf{X}\rvert$	Vektorin $\mathbf{X}$ elementtien määrä
$\mathbf{X}^n$	n elementtiä sisältävä vektori $\mathbf{X}$
$\mathbf{X}_i$	Vektorin $\mathbf{X}$ elementti i
$\mathbf{X}_{[a,b]}$	Vektorin $\mathbf{X}$ elementit suljetulla välillä elementistä a elementtiin b
$\mathbf{X} \leftarrow \mathbf{R}\{0,1\}^n$	Vektori $\mathbf{X}$ on tasaisen jakauman mukaan satunnaisesti valittu joukon $\{0,1\}^n$ alkio
$gcd(a,b)$	Suurin yhteinen tekijä. Luku, joka jakaa molemmat luvut a ja b siten, että lopputulos on kokonaisluku
$a \, mod \, n$	a modulo n. Jakojäännös, kun luku a jaetaan luvulla n
$\oplus$	Poissulkeva-tai operaatio, XOR
$a \equiv b \, (mod \, n)$	Luvut a ja b ovat kongruentteja modulo n. Luvut a' ja b' ovat kongruentteja, jos niiden jakojäännökset n:llä jaettaessa ovat samat
$\approx$	Lähes yhtä suuri kuin vertailuoperaattori
$\bullet$	Elliptisellä käyrällä tapahtuva kertolasku operaatio

1. JOHDANTO

Internet ja elektroniset transaktiot, kuten esimerkiksi tilisiirrot ja sähköiset hakemukset, ovat mullistaneet palveluita ja ihmisten tapaa asioida. Internetin kehittymisen myötä suuri osa palveluista ja rahaliikenteestä ovat siirtyneet internetiin, ja tarve kasvokkain käytävälle asioinnille on vähentynyt. Palveluntarjoajat ovat voineet yhdistää palveluitaan, ja on syntynyt uusia nopeasti kasvavia yrityksiä, jotka tavoittavat joukoittain ihmisiä.

Samalla kun yrityksillä on mahdollisuus tarjota monipuolisia palveluita edullisesti ja globaalisti, on noussut huoli siitä, miten merkittävässä asemassa olevat yritykset toimivat vastuullisesti, ja tarjoavat palveluitaan tasapuolisesti kaikille esimerkiksi maasta, varallisuudesta ja sosiaalisesta statuksesta riippumatta.

Kolmansien osapuolten tarjoamien palveluiden uhkakuvina voidaan nähdä muun muassa yksityisen datan kerääminen [1], keskittynyt päätösvalta, mahdollisuus valita asiakkaat tai asiakkaiden jääminen ulkopuolelle [2], toiminnan läpinäkymättömyys sekä palveluiden hitaus ja maksullisuus. Myös tietomurrot keskitetyillä palvelimilla ylläpidettäviin tietokantoihin uhkaavat koko käyttäjäkunnan omaisuutta ja yksityisyyttä [3].

Ensimmäinen kuvaus lohkoketjuista, "Bitcoin: A Peer-to-Peer Electronic Cash System" julkaistiin vuonna 2008. Paperissa esitellään elektroninen raha, joka mahdollistaa osapuolten välillä lähetettävät maksut ilman keskitettyä palvelinta tai hallintoa [4]. Lohkoketjuteknologia toteuttaa ratkaisun hajautettuun aikaleimaukseen sekä hajautetun järjestelmän konsensukseen. Lohkoketjuteknologian etuna voidaan nähdä läpinäkyvät, toimintavarmat sekä viiveettömät sovellukset. Sovelluksia on kehitetty muun muassa arvon siirtämiseen, omistussuhteiden kuvaamiseen ja identiteetin hallintaan.

Lohkoketjujen toimintaperiaatetta kuvaa termi *keskittämätön tilikirja*, joka juontuu siitä, että kaikki verkon osapuolet yhdenarvoisesti pitävät kirjaa kaikista transaktioista. Jokaisen kirjanpitäjän tulee lisätä samat transaktiot tilikirjaan, jotta tilikirjan osoittamasta tilasta voidaan olla yhteisesti yksimielisiä. Jokainen transaktio tilikirjassa on digitaalisesti allekirjoitettu.

Transaktioiden digitaalinen allekirjoitus tapahtuu tilikohtaisella yksityisavaimella. Avaimella allekirjoitettu transaktio on ainut tapa todistaa tilikohtaisia tapahtumia lohkoketjuissa. Transaktion allekirjoituksesta voidaan laskea tilin tunnistamiseen tarkoitettu julkinen tietue, josta ei puolestaan ole mahdollista laskea yksityisavainta. Jokainen transaktio on kaikkien julkisesti nähtävillä. Yksityisyys lohkoketjuissa menetetään, jos julkinen avain on yhdistettävissä henkilökohtaisiin tietoihin.

Jos tilikohtainen yksityinen avain kadotetaan, ei ole mitään tapaa tehdä transaktioita kyseiseltä tililtä. Puolestaan, jos tilikohtainen yksityinen avain paljastuu toiselle käyttäjälle, toisella käyttäjällä on samat mahdollisuudet tehdä tilikohtaisia transaktioita kuin alkuperäisellä omistajalla, esimerkiksi siirtää varat kokonaan hallitsemalleen tilille.

Varojen ja yksityisyyden suojaamiseksi voidaan tehdä useita tilejä, jotta millään yksittäisellä tilillä ei ole merkittävän suurta rahallista arvoa ja jolloin

myös motivaation tilin kaappaamiselle, etenkin kohdistetulle hyökkäykselle, katsotaan olevan pienempi.

Avaimen generointiin, käyttämiseen ja säilyttämiseen on kehitetty kryptolompakko (crypto wallet). Käyttäjä voi valita käyttötarkoitukseen sopivan lompakon hallitakseen henkilökohtaisia tilejään. Kryptolompakon turvallinen toiminta on ratkaisevaa käyttäjän yksityisyyden ja tilin turvallisuuden kannalta.

Tässä diplomityössä toteutetaan tietoturvallinen avainsäilö suojattuun ohjelmaympäristöön, ja esitetään kryptografinen menetelmä, jonka avulla voidaan vähentää tilin kadottamisen riskiä turvallisuudesta tinkimättä. Menetelmää voidaan käyttää turvaamaan myös muussa kuin henkilökohtaisessa käytössä olevia kryptoavaimia, esimerkiksi lohkoketjuverkossa toimivien IoT-laitteiden (Internet of Things, esineiden internet), Edge-laitteiden tai autonomisten autojen yhteydessä.

2. KRYPTOGRAFIA

Tietoturvan avulla voidaan turvata käsiteltävien tietojen luottamuksellisuus, eheys ja saatavuus. Kryptografia on tärkeä työkalu, jolla tietoturvaa pyritään saavuttamaan.

Tiedon salauksella on ollut tärkeä rooli viestien sisällön turvaamiseksi jo tuhansien vuosien ajan [5]. Uusien salaustekniikoiden kehittäminen ja niiden murtaminen ovat vuorotelleet jo ensimmäisistä salausmenetelmistä lähtien. Tiedon luottamuksellisuuden, alkuperän ja muuttumattomuuden todistaminen on muuttunut merkittävästi, kun on siirrytty allekirjoitetuista papereista ja sinetöidyistä kirjekuorista digitaalisessa muodossa lähetettäviin viesteihin. Salausta on alun perin käytetty etenkin turvaamaan sotilaallisia ja diplomaattisia viestejä, mutta nykypäivän digitaalisessa yhteiskunnassa kryptografia on tärkeä osa jokapäiväistä tiedonsiirtoa.

Unohtamatta matematiikan ja sähkötekniikan tutkimuksen merkitystä, tietojenkäsittelytieteiden kehittyessä moderni kryptografia mahdollistaa tiedon salauksen, digitaalisen allekirjoittamisen sekä avaimenvaihdon tarkoituksellisen hitaasti murrettavilla, tai joissakin tapauksissa todistettavasti turvallisilla menetelmillä.

Kappaleessa esitellään tässä työssä käytetyt kryptografiset menetelmät, joita ovat hash-funktio, digitaalinen allekirjoitus, salaisuuden paloittelu sekä epäsymmetrinen salaus.

2.1. Hash-funktio

Kryptografisen hash-funktion (cryptographic hash function) tehtävänä on tiivistää mille tahansa viestille eli alkukuvalle yksilöllinen vakiomittainen hajautusarvo. Hajautusarvon vakiomitta voi olla esimerkiksi 256-bittiä, jolloin kahden pitkän viestin sisältö voidaan todeta poikkeavaksi tai täsmälleen samaksi pelkästään viesteistä koostettuja lyhyitä tiivisteitä vertailemalla. Kryptografinen hash-funktio toteuttaa seuraavat ominaisuudet [6]:

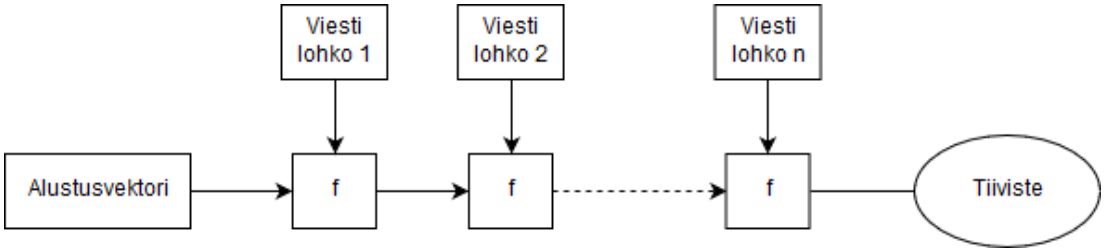
1. *Alkukuva resistentti* (preimage resistance), hajautusarvon y avulla ei ole laskennallisesti tehokasta tapaa löytää viestiä M, joka tiivistyy samaksi hajautusarvoksi $hash(M) = y$

2. *Toinen alkukuva resistentti* (second preimage resistance), annetulle alkukuvalle $M1$ ei ole laskennallisesti tehokasta tapaa löytää vastaavaa alkukuvaa $M2$, joka täyttäisi ehdon $hash(M1) = hash(M2)$

3. *Yksilöllisyys* (collision resistance), ei ole laskennallisesti tehokasta tapaa löytää mitään kahta alkukuvaa $M1$ ja $M2$, jotka tiivistyvät samaksi hajautusarvoksi $hash(M1) = hash(M2)$

Kuvassa 1 esitellään Merkle-Damgård rakenteeseen perustuva hajautusarvon laskenta. Merkle-Damgård muodostuksessa pakkausfunktioon f syötetään ensimmäisellä kierroksella alustusvektori sekä ensimmäinen viestilohko.

Ensimmäisen viestilohkon jälkeen pakkausfunktiolle syötetään iteratiivisesti seuraava viestilohko ja aiempi pakkaustulos. Viimeinen viestilohko täydennetään lohkon monikerran mittaiseksi ja viimeiseltä pakkausfunktiolta tulokseksi saadaan määrämittainen tiiviste.



Kuva 1. Merkle-Damgård tiivisteen muodostus

Edellä kuvattu menetelmä on yksinkertainen, mutta ei kryptografisesti turvallinen tapa tuottaa hajautusarvo, sillä menetelmää vastaan tunnetaan hyökkäyksiä [7]. Esimerkiksi Ethereum transaktioiden hajautusarvot muodostetaan Keccak256 hash-funktion avulla [8]. Keccak-algoritmin tuottamat tiivisteet perustuvat sponge-muodostukseen [9], jolle ei tunneta vastaavia hyökkäyksiä kuin Merkle-Damgård muodostuksen avulla tuotetuille tiivisteille.

Hash-funktioita käytetään muun muassa seuraavissa käyttökohteissa:

- Digitaalinen allekirjoitus
- Tietorakenteet, hajautustaulut
- Tiedon muuttumattomuuden ja identtisyyden vertaileminen
- Tiedon olemassaolon etukäteen todistaminen
- Työntodistus algoritmi

2.2. Työntodistus protokolla

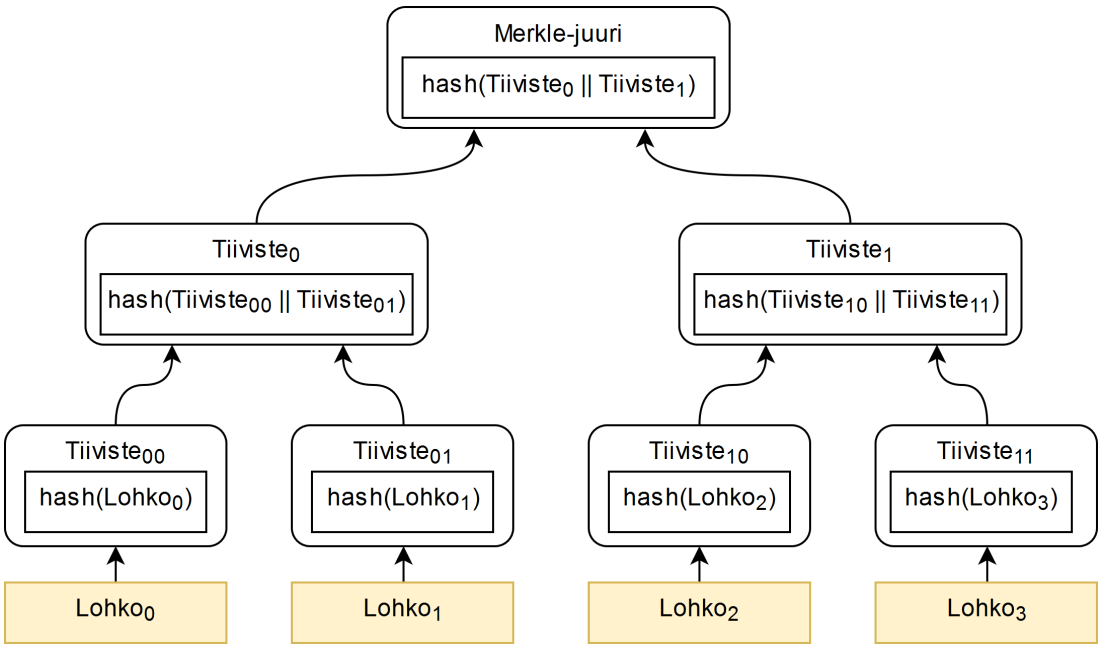
Työntodistus protokolla (PoW, Proof of Work) on kehitetty rajoittamaan yleisesti resurssien käyttöä edellyttämällä resurssin pyytäjää ratkaisemaan laskentatehoa vaativa, mutta helposti tarkistettavissa oleva tehtävä [10]. Ensimmäinen tunnettu sovellus, joka käyttää edellä kuvattua menetelmää, on Hashcash [11]. Hashcashin kustannusfunktiona toimii alkukuvan etsiminen, jonka hajautusarvo alkaa ennalta sovitulla määrällä nollia. Alkukuva koostuu ennalta määritellystä *otsikosta* ja ratkaistavasta *noncesta*. Kustannusfunktio voi olla yleisesti tiedossa oleva, jolloin otsikkona voi toimia esimerkiksi palvelunnimi, tai se voi olla vuorovaikutteinen, jolloin otsikkoon lisätään nimen lisäksi palveluntarjoajalta pyydetty kertakäyttöinen haaste. Turvallista hash-funktiota käytettäessä nopein keino ratkaista tehtävä on suorittaa hash-funktiota eri nonceilla kunnes löytyy alkukuva, jonka hajautusarvo sisältää vaaditun määrän etunollia. Kustannusfunktion ratkaisun tarkistaminen tapahtuu vastaavasti laskemalla alkukuvalle hajautusarvo ja katsomalla alkaako hajautusarvo vaaditulla määrällä nollia.

Ei-vuorovaikutteisen työntodistuksen ongelmana voidaan nähdä ennakkovalmisteltu hyökkäys (pre-computation attack). Ennalta määriteltyihin otsikoihin on mahdollista laskea etukäteen ratkaisuja ja aiheuttaa ennakkovalmisteltu lyhytaikainen palvelunestohyökkäys. Myöskään vuorovaikutteista työntodistusta ei pidetä toteuttamiskelpoisena jokaiseen ympäristöön, esimerkiksi sähköpostipalveluihin. Yksi keino rajoittaa ennakkovalmisteltuja hyökkäyksiä on sisällyttää otsikkoon ennustamaton ja vaihtuva, yleisesti tunnettu muuttuja, kuten esimerkiksi viikon lottonumerot [11].

2.3. Merkle-puu

Merkle-puu on tietorakenne, joka mahdollistaa useiden datalohkojen tiivistämisen yhdeksi hajautusarvoksi [12]. Puurakenteessa lehdet, eli alimmaiset solmut, ovat yksittäisten lohkojen hajautusarvoja ja kaikki loput solmut sisältävät alempien solmujen yhteisen hajautusarvon. Merkle-puun hyötynä on, että minkä tahansa yksittäisen lohkon sisältö voidaan tarkistaa tuntematta muiden lohkojen sisältöä.

Kuvan 2 esimerkkitilanteessa, jossa tiedostoa ladataan palvelimelta, lohko_0:n sisällön eheys voidaan tarkastaa tuntemalla Merkle-juuri, tiiviste_{01} sekä tiiviste_1.



Kuva 2. Merkle-puu

2.4. Julkisen avaimen kryptografia

Epäsymmetrisellä- eli julkisen avaimen salauksella viitataan algoritmeihin, joissa salaukseen käytetään eri avainta kuin salauksen purkamiseen. Epäsymmetrisen salauksen vastakohta on symmetrinen salaus, jossa sama salausavain käy sekä purkamiseen että salaamiseen. Etuna symmetriseen salaukseen on, että lähettäjän ja vastaanottajan ei tarvitse sopia käytettävää salausavainta etukäteen ennen ensimmäistä lähetettyä viestiä, vaan viestin vastaanottaja voi julkaista julkisen avaimen Pk (public key) useille osapuolelle paljastamatta purkamiseen vaadittavaa avainta Sk (secret key). Tiedon salaamisen lisäksi, julkisen avaimen kryptografia mahdollistaa digitaalisen allekirjoittamisen.

2.5. RSA-salaus

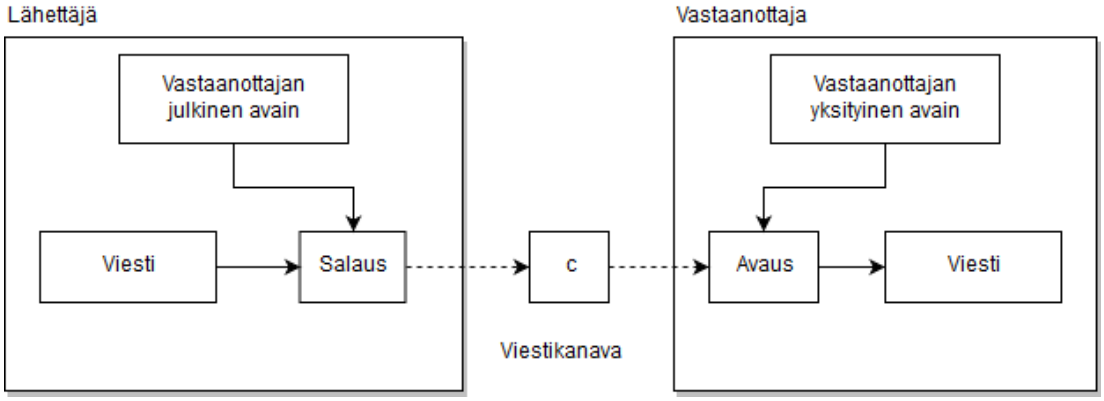
RSA (Rivest-Shamir-Adleman) on ensimmäinen julkaistu käytännöllinen epäsymmetrisen kryptografian salausmenetelmä [13]. RSA määrittelee salaovifunktion, jota voidaan käyttää tiedon salaamiseen ja allekirjoittamiseen.

Funktiota f sanotaan salaovifunktioksi (trapdoor function), jos $c = f(m, e)$ on helposti laskettavissa tunnetuilla parametreilla m ja e samalla, kun $m = f^{-1}(c, d)$ ratkaiseminen on vaikeaa tunnetulla parametrillä c ilman salaista parametria d. RSA:n määrittelemä salaovifunktio perustuu siihen että annettujen suurten lukujen tulo on nopeasti laskettavissa, mutta tekijöiden löytäminen on huomattavan hankalaa [13].

RSA:n julkinen avain koostuu lukuparista (e, n) ja yksityinen avain lukuparista (d, n), jotka muodostetaan seuraavasti:

1. Valitaan kaksi satunnaista alkulukua $p \in \mathbb{P}$ ja $q \in \mathbb{P}$ siten, että binäärimuodossa luvut ovat keskenään lähes samanpituisia $|p| \approx |q|$.
2. Lasketaan $n = p \cdot q$.
3. Lasketaan $\phi(n) = (p-1) \cdot (q-1)$.
4. Valitaan satunnaisluku $e \in \mathbb{N}_{>0}$, jossa $e < \phi(n)$ ja $gcd(e, \phi(n)) = 1$.
5. Ratkaistaan $d \,|\, d \in \mathbb{N}_{>0}$ siten, että $e \cdot d \equiv 1 \; (mod\, \phi(n))$.

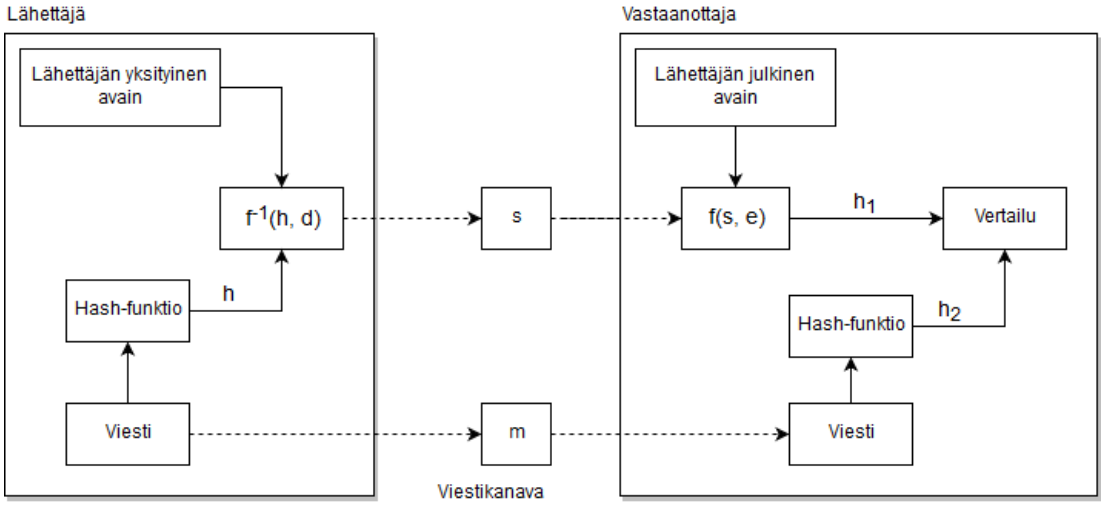
Salattava viesti sisällytetään lohkoon, jonka pituus on korkeintaan $log_2(n)$ bittiä. Kryptaukseen käytetään kaavaa $c = (m^e) \; mod \; n$, kun puolestaan purkamiseen kaavaa $m = (c^d) \; mod \; n$, joissa m viittaa salattavaan lohkoon, ja c viittaa salattuun lohkoon. Kuvassa 3 esitellään RSA-salaus.



Kuva 3. RSA salausoperaatio

Salausfunktiosta $c = (m^e)\ mod\ n$ käytetään salauksen yhteydessä jatkossa merkintää $c = E(m, pk)$, jossa m:llä viitataan salattavaan viestiin ja pk:lla salausavaimeen. Purkufunktiosta salauksen yhteydessä käytetään vastaavasti merkintää $m = D(c, sk)$, jossa c on avattava viesti, ja sk purkamiseen vaadittava avain.

RSA:n purkufunktiota f^{-1} voidaan käyttää myös viestien allekirjoittamiseen, Kuva 4. Menetelmässä lasketaan viestin m hajautusarvo $h = hash(m)$. Allekirjoituksessa lähettäjän yksityistä avainta d käyttäen saadaan todiste $s = f^{-1}(hash(m), d)$. Vastaanottaja käyttää lähettäjän julkista avainta e ja RSA-salausfunktiota f saadakseen vertailuarvon $h_1 = f(s, e)$. Saapuneesta viestistä otetaan tiiviste $h_2 = hash(m)$ ja, jos $h_1 = h_2$ voidaan olla varmoja, että viestin m allekirjoittaja on käyttänyt julkisen avaimen e salaista paria d.



Kuva 4. Digitaalinen allekirjoitus

Purkufunktiosta $s = f^{-1}(h, d)$ käytetään allekirjoitusoperaation yhteydessä jatkossa merkintää $s = sign(o, sk)$, jossa o:lla viitataan allekirjoitettavaan otsikkoon ja sk:lla lähettäjän yksityisavaimeen d.

2.6. Elliptisten käyrien kryptografia

Elliptisten käyrien salausmenetelmät (ECC, Elliptic Curve Cryptography) ovat julkisen avaimen salausmenetelmiä, jotka liittyvät elliptisellä käyrällä tehtäviin laskutoimituksiin. Turvallisuus perustuu diskreetin logaritmin ongelmaan elliptisellä käyrällä (ECDLP, Elliptic Curve Discrete Logarithm Problem) [14][15]. Elliptisten käyrien kryptografiaa voidaan käyttää myös viestien allekirjoittamiseen (ECDSA, Elliptic Curve Digital Signature Algorithm).

Toisin kuin kertolaskun tekijöiden löytämiseen, ECDLP:n ratkaisuun tunnetaan ainoastaan eksponentiaalisen ajan vaativia algoritmeja. Tästä johtuen elliptisten käyrien kryptografialla on mahdollista saavuttaa sama suojaustaso lyhyemmillä avainpituuksilla, verrattuna muihin julkisen avaimen menetelmiin. Esimerkiksi ECC-160 avain mahdollistaa vastaavan turvallisuuden RSA-1024 avaimeen verrattuna, samalla kun ECC-224 tarjoaa verrannollisen turvallisuuden RSA-2048 avaimeen nähden [16].

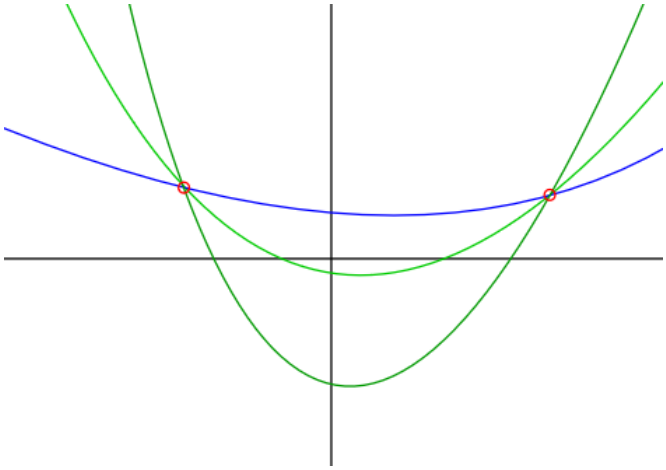
2.7. Salaisuuden paloittelu

Salaisuuden jakamisella tai paloittelulla tarkoitetaan menettelyä, jossa salaisuus S jaetaan n osaan X_n siten, että mikään yksittäinen $X_i \mid i \leq n, i \in \mathbb{N}_{>0}$ ei paljasta informaatiota salaisuudesta. Yksinkertaisin esimerkki on salaisuuden S jakaminen salatekstiin ja avaimeen X ja Y, operaatiolla $Y = S \oplus X \mid X \leftarrow R\{0,1\}^{|S|}$, jossa R on kryptografisesti turvallinen pseudosatunnaislukugeneraattori, ja $|S|$ merkitsee salaisuuden pituutta. Salaisuuden palauttaminen tapahtuu operaatiolla $S = X \oplus Y$. Taustalla käytettävää salausmenetelmää kutsutaan One-time padiksi (OTP) ja oikein käytettynä sen tuottamaa salatekstiä on mahdotonta purkaa [17][18].

One-time pad ei kuitenkaan suoraan sovellu salaisuuden jakamiseen useampaan osaan. Erityisesti tilanteessa, jossa salaisuuden halutaan olevan palautettavissa $k \mid 0 < k \leq n$ määrällä avaimia, tarvitaan edistyneempiä menetelmiä. Yleisnimitys edellä mainitulle on (k, n)-kynnysarvomenetelmä ((k, n)-treshold scheme).

Tunnetuin kynnysarvomenetelmä on Shamirin salaisuuden jako (SSSS, Shamir Secret Sharing Scheme) [19], jossa salaisuus S on $k-1$ asteisen polynomin arvo pisteessä 0 ja osatiedot ovat polynomin arvoja muissa pisteissä. Osatietojen muodostaminen tapahtuu valitsemalla arvot n ja k, sekä valitsemalla alkuluku $p \mid p > n, p > S$. Generoimalla satunnaisluvut $a_1, a_2, \ldots, a_{k-1} \mid a_i < p, a_i \in \mathbb{N}_{>0}$, jonka jälkeen muodostamalla polynomi $f(x) = S + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$. Lopulta salaisuus jaetaan n osaan kaavalla $X_i = (i+1, f(i+1) \bmod p) \mid 0 \leq i < n$. Salaisuuden palauttaminen osatiedoista tapahtuu Lagrangen interpolaatiopolynomin [20] avulla.

Kuvassa 5 havainnollistetaan, miten kahden annetun pisteen päälle voidaan piirtää ääretön määrä kaksiasteisia polynomeja. Kaksiasteisen polynomin määrittelyyn yksikäsitteisesti vaaditaan kolme pistettä. Kuvasta poiketen Shamirin salaisuuden jaossa käytettävät polynomit määritellään kaksiulotteisen tason sijaan äärellisessä kunnassa.



Kuva 5. Kaksiasteisia polynomeja, jotka leikkaavat kaksi pistettä kaksiulotteisella tasolla

2.8. Avaimen muodostus funktio

Algoritmia, joka tuottaa yhdestä salasanasta, salasanalauseesta tai avaimesta yhden tai useamman avaimen kutsutaan avaimen muodostus funktioksi (KDF, Key Derivation Function). Funktiota voidaan käyttää muun muassa muodostamaan salasanasta pidempi satunnaisavain lohkosalaimeen, muodostamaan useita salasanoja yhdestä avaimesta tai muuttamaan salasana toiseen formaattiin. KDF-funktio on deterministinen ja yksisuuntainen, eli tuottaa samalle syötteelle aina saman lopputuloksen eikä lopputuloksesta voida päätellä syötettä.

Lohkoketjukontekstissa BIP39-standardi [21] (Bitcoin Improvement Proposal 39) määrittelee avaimen muodostus funktion yhteydessä käytettävän muistifraasin. Menetelmän tarkoituksena on muuttaa tietokoneen luoma satunnaisuus ihmiselle helpommin luettavaan sanalliseen muotoon. Standardi määrittelee englanninkielisen sanalistan, josta muistifraasin muodostamiseksi, valitaan satunnaisesti sanoja. Muistifraasista johdetaan lompakon muodostamiseen vaadittava avain PKDF2-funktion (Password-Based Key Derivation Function 2) [22] avulla. Muistifraasi on yksityisenä pidettävä tietue, josta avaintietue voidaan palauttaa. Standardissa esitellään myös menetelmä, jossa muistifraasin lisäksi vaaditaan salasana avaintietueen palauttamiseksi.

3. KESKITTÄMÄTTÖMÄT JÄRJESTELMÄT

Hajautetulla järjestelmällä tarkoitetaan yhtenä kokonaisuutena ilmenevää järjestelmää, jossa noodit käsittelevät tietoa. *Keskittämätön järjestelmä* on sellainen hajautettu järjestelmä, jossa yksikään noodi ei määrittele toisten noodien toimintaa, eikä järjestelmä ole riippuvainen minkään yksittäisten noodien toiminnasta. *Keskitetty järjestelmä* voi puolestaan olla hajautettu tai *hajauttamaton järjestelmä*. Erona keskittämättömään järjestelmään on, että keskitetyssä järjestelmässä on sellaisia yksittäisiä noodeja, joista järjestelmän tila on riippuvainen [23].

3.1. Bysanttilainen ongelma

Bysanttilaisen ongelman esittivät vertauskuvallisesti ensimmäisen kerran Akkoyunlu, Ekanadham ja Huber [24] sekä myöhemmin paremmin tunnetussa muodossa ongelman esittivät Lamport, Shostak ja Pease [25]. Jälkimmäisessä versiossa kuvataan millä ehdoilla joukko kenraaleita voivat muodostaa konsensuksen yhteisestä hyökkäyssuunnitelmasta vihollisen kukistamiseksi samalla, kun kenraalien joukossa on juonittelevia epärehellisiä kenraaleita, joiden tavoitteena on aiheuttaa sekasortoa ja rehellisten kenraalien yhteisen hyökkäyksen epäonnistuminen.

Paperissa esitetään, että kaksi rehellistä kenraalia eivät voi päästä yhteisymmärrykseen lähetin välityksellä suullista viestiä vaihtamalla, jos heidän lisäksi joukossa on yksikin epärehellinen kenraali. Puolestaan, kun joukossa on yli kolme kenraalia, paperissa kuvattua menetelmää käyttäen rehelliset kenraalit päätyvät yhteisymmärrykseen, kun korkeintaan kolmasosa joukosta on epärehellisiä [25].

Hajautetuissa järjestelmissä bysanttilaisella ongelmalla kuvataan yleisesti tilannetta, jossa järjestelmän komponentit voivat toimia väärin, määrittelemättömällä viiveellä ja ilman täydellistä informaatiota, onko jokin järjestelmän komponentti pettänyt. Keskittämättömässä järjestelmässä, kuten lohkoketjut, täytyy määritellä algoritmi, jonka avulla on mahdollista päästä yhteisymmärrykseen järjestelmän nykyisestä tilasta.

3.2. Aikaleimaus

Tilanteessa, jossa dokumentin d olemassaolo ajanhetkellä t täytyy pystyä todistamaan jälkeenpäin ajanhetkellä $t + 1$, voidaan käyttää aikaleimausta. Yksinkertaisin ratkaisu on keskitetty aikaleimapalvelu (TSS, Time Stamping Service). Ratkaisussa asiakas toimittaa dokumentin aikaleimapalveluntarjoajalle, joka puolestaan kirjaa dokumentin vastaanottoajan sekä säilyttää dokumenttia siltä varalta, että alkuperäisen dokumentin olemassaolo jollakin ajanhetkellä kyseenalaistetaan. Menetelmä ei kuitenkaan tarjoa yksityisyyttä eikä ole turvallinen, vaan todistaminen siirretään luotetulle toimijalle [26].

Yksityisyyttä voidaan lisätä tarjoamalla palveluntarjoajalle dokumentista vain hajautusarvo $y = hash(d)$. Myös todiste s luotetun kolmannen osapuolen aikaleimasta voidaan siirtää takaisin asiakkaalle sen jälkeen, kun TSS lisää hajautusarvoon ajanhetken t ja allekirjoittaa otsikon $o = (y \| t)$ käyttäen palveluntarjoajan yksityistä avainta sk_{tss}, $s = sign(o, sk_{tss})$. Menetelmän avulla palveluntarjoajan ei tarvitse nähdä dokumenttia eikä säilyttää todistetta, vaan dokumentin olemassaolon ajanhetkellä t todistamiseen riittää allekirjoitettu todistus, alkuperäinen dokumentti, aikaleima sekä TSS:n julkisen avaimen tunteminen. Menetelmä vaatii edelleen luottamuksen palveluntarjoajaan, ettei se laadi varhaistettuja aikaleimoja [26].

Suhteellinen aikajärjestys on mahdollista saavuttaa sisällyttämällä otsikkoon o_n edellisen otsikon tiiviste, jota selkeyden vuoksi kutsutaan linkitysarvoksi $l_n = hash(o_{n-1})$. Vuoronumeroltaan n:s todiste lasketaan nyt $s_n = sign(o_n, sk_{tss}) = sign(y_n \| t_n \| l_n, sk_{tss})$. Menetelmän avulla voidaan todistaa, ettei varhaisemman tuntemattoman dokumentin todistusta s_{n-a} ole luotu myöhemmin kuin tunnetun dokumentin todistus s_{n-b}, jos tunnetun dokumentin aikaleima ja linkitysarvot $[l_{n-a}, l_{n-b}]$ tuntemattomasta dokumentista tunnettuun dokumenttiin tunnetaan.

Hajautetun aikaleimauksen avulla on mahdollista luoda usean toimijan yleisesti tunnettu linkitetty aikaleimapuu [27]. Nykyaikainen toteutus hajautetusta aikaleimapalvelusta on lohkoketju.

3.3. Sybil-hyökkäys

Erityisesti verkoissa, joissa osapuolet eivät tunne toisiaan, on mahdollista, että yhdellä käyttäjällä on hallussaan useita identiteettejä. Sybil-hyökkäyksessä hyökkääjä luo useita identiteettejä, joiden avulla hän pyrkii saamaan vaikutusvaltaa verkossa. Perinteisesti Sybil-hyökkäystä voidaan hidastaa vaatimalla käyttäjiä varmentamaan tunnus sähköpostin avulla, jolloin useiden identiteettien luomisen katsotaan olevan hitaampaa. Keskittämättömässä verkossa käyttäjien varmentaminen on vaikeaa ja se mahdollisesti heikentäisi käyttäjien yksityisyyttä.

4. LOHKOKETJUTEKNOLOGIA

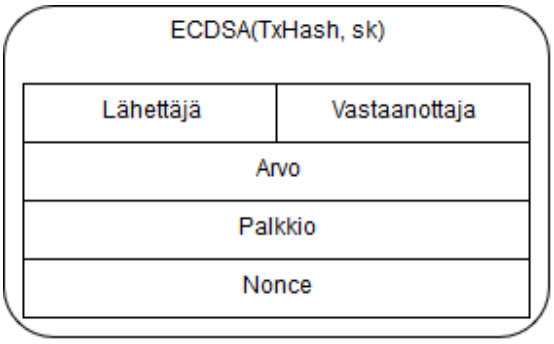
Lohkoketjut ovat uusi, kehittyvä ja laaja käsite. Tässä työssä lohkoketjuteknologiaa kuvataan ensimmäisten näkyvien ja tällä hetkellä yleisimpien lohkoketjujen, Bitcoinin ja Ethereumin näkökulmasta. Teknologiat, kuten digitaalinen allekirjoitus, hajautetut verkot, aikaleimaus sekä työntodistus, ovat toimineet lohkoketjun mahdollistajina. Kuitenkin nimenomaan lohkoketjuteknologiat määrittelevät ensimmäisenä julkisen käyttäjien hallitseman keskittämättömän tilikirjan.

Ensimmäinen käytännön lohkoketjusovellus on Bitcoin, joka määritteli valuutan sekä säännöt siitä, mikä on valuutan alkuperäinen liikkeelle laskettu määrä ja miten valuuttaa lasketaan liikkeelle eri lohkoissa. Lohkoketjujen sovellutuksia ei kuitenkaan kannata rajata pelkästään valuuttaan, vaan teknologiana lohkoketju tarjoaa alustan, jota seuraavat ominaisuudet kuvaavat [28]:

- Jaettu tapahtumarekisteri
- Tietojen muuttumattomuus
- Hajautettu vikasietoisuus
- Kryptografinen suojaus
- Tapahtumien luotettavuus ja auditoitavuus
- Automaattisuus ja riippumattomuus
- Tapahtumien ainutkertaisuus
- Tunnistettavuus
- Toimintojen ajantasaisuus ja oikeellisuus

4.1. Peruuttamattomat transaktiot

Lohkoketjujen lähtökohtana on kaikille osapuolille jaetut, digitaalisesti allekirjoitettavat ja varmennettavat transaktiot. Kuva 6 esittelee transaktion yleistetyn rakenteen sisältäen lähettäjän osoitteen, vastaanottajan osoitteen, transaktion arvon, transaktiopalkkion, noncen sekä lähettäjän digitaalisen allekirjoituksen.



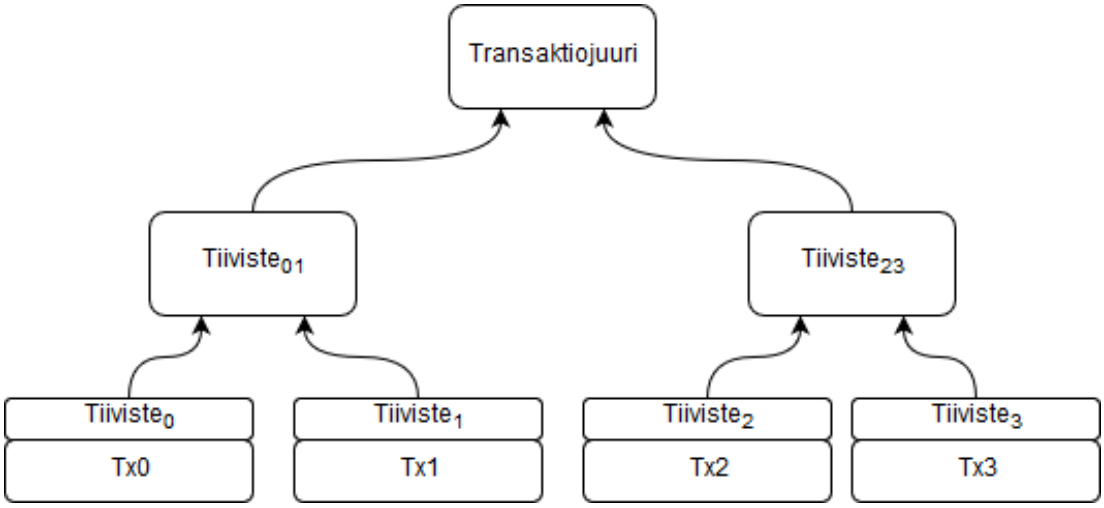
Kuva 6. Transaktio

Noncen tehtävänä on estää saman transaktion uudelleen julkaiseminen (replay-attack), joko tilikohtaisen järjestysnumeron avulla tai viittaamalla tilillä tehtyyn edelliseen transaktioon. *Arvolla* viitataan kryptovaluuttaan (cryptocurrency), joka siirretään *lähettäjän* osoitteesta *vastaanottajan* osoitteeseen.

Lähettäjä maksaa julkaisemastaan transaktiosta *palkkion*. Palkkion määrä pääsääntöisesti ratkaisee, mitkä transaktiot louhija valitsee julkaistavaan lohkoon. Lohkossa julkaistavien transaktioiden määrä on rajallinen, joten palkkio rajoittaa verkkoa ylikuormittumiselta. Kun lohkoketjuun tulee paljon uusia transaktioita, käyttäjät joutuvat maksamaan enemmän palkkiota saadakseen oman transaktion julkaistavaan lohkoon.

Lisäksi transaktioissa voi olla *data*-tietue, joka sisältää monimuotoisemman arvon. Esimerkiksi Ethereum-transaktioissa datakenttä sisältää ohjelmakoodia tai funktiokutsuja.

Lohkossa transaktiot on järjestetty Merklen puuhun, joka mahdollistaa yksittäisten transaktioiden nopean varmistamisen latamaatta kaikkia lohkoon liittyviä transaktioita. Kuvassa 7 esitetään miten transaktioista $Tx0$, $Tx1$, $Tx2$ ja $Tx3$ koostetaan transaktiojuuri.



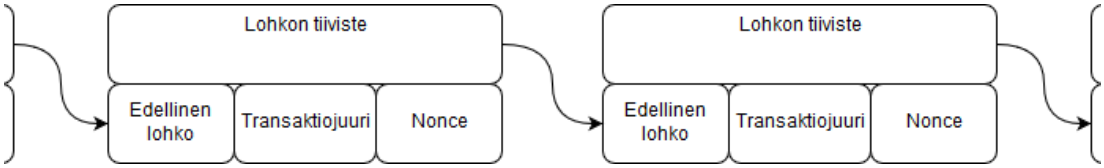
Kuva 7. Transaktiojuuri

Järjestämätön lista transaktioista ei kuitenkaan kuvaisi yksiselitteisesti hajautetun järjestelmän tilamuutosta eikä mahdollistaisi esimerkiksi saman rahan kaksinkertaisen kuluttamisen estämistä. Transaktioiden järjestyksen ja konsensuksen saavuttamiseksi tarvitaan yhteinen kello, hajautettu aikaleimaserveri. Bitcoin yhdistää Hashcashissa [11] käytetyn kaltaisen työntodistus menetelmän, aiemmin esitettyihin hajautettuihin aikaleimaratkaisuihin [26][27]. Näin saavutetaan täysin keskittämätön aikaleimaserveri, jossa jokaisella hajautetun verkon osapuolella on omasta laskentatehostaan riippuva mahdollisuus löytää uusi aikaleima eli lohko.

Uuden lohkon julkaiseva osapuoli määrittelee lohkossa viitatut transaktiot ja niiden järjestyksen. Pisin lohkojen muodostama ketju määrittelee järjestelmän nykyisen tilan. Aikaleimaserverin ominaisuuksien takia lohkojen järjestystä

ei voi muuttaa, luomatta vaihtoehtoisia aikaleimoja. Työntodistus tekee transaktiohistorian uudelleenkirjoitusoperaatiosta laskennallisesti raskaan.

Tietokanta on jaettu järjestyksessä viitattuihin lohkoihin, joista jokaiseen yksittäiseen lohkoon on listattu siihen liittyvät tilanmuutokset, transaktiot. Jokainen uusi lohko viittaa edelliseen lohkoon siten, että minkään edeltävän lohkon muokkaus ei onnistu luomatta myös kaikkia sen jälkeisiä lohkoja uudestaan. Kuvassa 8 on yleistetty lohkoketjujen rakenne. Esimerkiksi Bitcoin-lohkon otsikossa olisi lisäksi versio-, aikaleima- ja kustannusfunktion tavoitevaikeustaso-tietueet.



Kuva 8. Lohkoketju

Lohkojen koko ja määrä on rajoitettu varmistaen, että kaikilla verkon osapuolilla on mahdollista tallettaa ja tarkastaa jokainen transaktio, vaikka tietokoneen resurssit olisivat rajalliset.

Konkreettisesti lohkoketju ilmenee käyttäjälle sovelluksena. *Lohkoketjusovellus* määrittelee säännöt alkutilalle ja hyväksytyille tilamuutoksille, joita se tarkkailee ja noudattaa. Jokainen sovellus pitää itsenäisesti kirjaa hyväksytysti tapahtuvista tilamuutoksista sekä niiden järjestyksestä. Kun jokaisen loppukäyttäjän sovelluksella on tasapuolinen mahdollisuus julkaista tilamuutos, joka hyväksymishetkellä jaetaan kaikkien verkkoon liittyneiden sovellusten kanssa samassa järjestyksessä, puhutaan lohkoketjuista. Toisin sanoen lohkoketju on useiden käyttäjien ylläpitämä keskittämätön tietokanta.

Lohkoketjujen periaatteellinen toiminta kuvataan seuraavasti [4]:

1. Uudet transaktiot julkaistaan kaikille käyttäjille
2. Jokainen osapuoli kokoaa transaktiot lohkoon
3. Jokainen osapuoli työskentelee löytääkseen työntodistuksen omalle lohkolleen
4. Kun työntodistus löydetään, lohko julkaistaan kaikille verkon osapuolille
5. Osapuolet hyväksyvät lohkon ainoastaan, jos kaikki transaktiot ovat valideja ja kuluttamattomia
6. Osapuolet osoittavat hyväksynnän julkaistulle lohkolle alkamalla louhimaan seuraavaa lohkoa, viitaten julkaistun lohkon tiivisteeseen edellisenä hyväksyttynä lohkona

4.2. Konsensusmenetelmä

Lohkoketjun käyttämä konsensusmenetelmä määrittelee, mitkä ja kenen julkaisemat lohkot kuvaavat lohkoketjun nykyisen tilan, eli keinoa jolla bysanttilainen ongelma ratkaistaan. Bitcoin ja Ethereum käyttävät

pääketjuissaan *Proof of Work* (PoW, työntodistus) konsensusalgoritmia. Muita konsensusalgoritmeja ja niiden variaatioita on lukuisia, mutta menetelmistä yleisimpiä ovat *Proof of Authority* (PoA) sekä *Proof of Stake* (PoS).

PoW-menetelmän kustannusfunktiona käytetään työssä aiemmin kuvattua sekä Hashcashissa käytettyä alkukuvan etsintää. Pisin sallittuja transaktioita sisältävä lohkoketju, jonka viimeinen lohko tiivistyy vaikeustason vaatimuksen mukaiseksi hajautusarvoksi, katsotaan kuvaavan järjestelmän tilan. Vaikeustaso on dynaaminen, ja siihen voidaan vaikuttaa määrämälllä työntodistuksen hajautusarvolle maksimiarvo. Vaikeustasoa säädetään siten, että lohkoaikaa pyritään pitämään vakiona.

PoA-menetelmässä voidaan käyttää listaa hyväksytyistä lohkon julkaisijoista, joista jokainen omalla vuorollaan, tai arvonnan perusteella saa julkaista uuden lohkon. Verkko on skaalautuva, ja se voidaan toteuttaa lyhyellä lohkoajalla. Haittapuolena on, että julkaisijoiden täytyy olla toisten julkaisijoiden tuntemia ja hyväksymiä. PoA:n avulla toteutetut lohkoketjut ovat jossain määrin keskitettyjä [29].

PoS-menetelmä käsittää toteutuksia, joissa louhintavuoro voidaan arpoa siten, että suuremmalla aktiivisuus- ja varallisuusmäärällä on suurempi mahdollisuus saada julkaista uusi lohko. Ethereum Serenity ehdottaa hyödynnettävän menetelmää, jossa verkon osapuolet saavat panttia vastaan osallistua lohkon julkaisuvuoron arvontaan. Menetelmässä hyväksytyn lohkon julkaisija saa lohkopalkkion, kun puolestaan hyväksymättömän lohkon julkaisija menettää pantin [30].

4.3. Enemmistöhyökkäys

Lohkoketjujen luottamus perustuu siihen, että yleisestä tilasta on olemassa yhteinen konsensus. Enemmistöhyökkäys kuvaa hyvin mitä voi tapahtua, jos yleinen konsensus menetetään. Tilanteessa, jossa hyökkääjän louhintakapasiteetti on tarpeeksi suuri, hyökkääjä voi onnistua korvaamaan aikaisemmin julkaistuja lohkoja julkaisemalla uuden pidemmän sarjan vaihtoehtoisia lohkoja. Hyökkääjällä on tällöin mahdollisuus valita vaihtoehtoisten lohkojen sisältämät transaktiot.

Hyökkäyksen motiivi perustuu siihen, että hyökkääjä tekee varoja kuluttavan transaktion, odottaa transaktion hyväksynnän ja lopulta julkaisee vaihtoehtoisen ketjun, jossa hänen transaktiotaan ei ole ollenkaan tapahtunut. Yli 50% louhintakapasiteetilla enemmistöhyökkäys onnistuu 100% todennäköisyydellä, koska hyökkääjä voi louhia vaihtoehtoista lohkoketjua nopeampaa kuin loppuverkko yhteensä. Alle 50% kapasiteetillakin enemmistöhyökkäys voi onnistua, mutta mitä pidempi vaihtoehtoinen ketju vaaditaan, sitä pienemmäksi onnistumisen todennäköisyys laskee.

Alle 50% hyökkäyksen hidastaminen on suoraviivaista: ennen kuin transaktio katsotaan tapahtuneeksi, odotetaan transaktion sisältävä lohko ja $n-1$ seuraavaa vahvistavaa lohkoa (confirmations). Mitä suurempi n on, sitä useamman lohkon hyökkääjä joutuu laskemaan ja odottamaan tehdäkseen korvaavan ketjun. Hyökkäyksen katsotaan olevan hyödytön koska kenelläkään ei odoteta olevan

ylivertaista louhintamekanismia eikä kapasiteettia, ja louhintapalkkioita pidetään suurempana hyötynä kuin hyökkäyksestä saatua etua [31].

Vaikka hyökkääjällä olisi ylivertainen louhintakapasiteetti muihin nähden, hyökkääjä ei kykene siirtämään valuuttaa muilta kuin hallitsemiltaan tileiltä tai luomaan valuuttaa tyhjästä. Myös luottamuksen menetys ja siitä seuraava valuutan arvon aleneminen voivat yleisesti rajoittaa lohkoketjuihin kohdistettavia massiivisia hyökkäyksiä.

4.4. Julkisuus

Lohkoketjuteknologian toimintaperiaatteen takia transaktiot ovat lohkoketjun osapuolien välillä lähtökohtaisesti julkista informaatiota, ja ne voivat sisältää käyttäjästä paljon informaatiota. Vaikka osoitteet ovat *pseudoanonyymejä*, yhdistelemällä käyttäjien tekemien transaktioiden tietoja voidaan osoite joissakin tapauksissa yhdistää todelliseen henkilöön.

Yksityisyyden suojelemiseksi käyttäjä voi luoda useita osoitteita ja käyttää useita hyppyjä rahansiirrossa tililtä toiselle. Tietojen näkyvyyttä voidaan rajata myös erilaisilla menetelmillä, kuten nollatietotodistuksella (zero knowledge proof) tai rajatuilla yksityisillä verkoilla.

4.5. Ethereum

Ethereum on Bitcoinin lisäksi toinen varhaisista ja nopeiten kasvaneista lohkoketjutoteutuksista. Muista lohkoketjuista poiketen Ethereum esitellään toimivan *keskittämättömien sovellusten* (DApp, decentralised applications) alustana [8], kun vastaavasti Bitcoin kuvataan keskittämättömänä rahajärjestelmänä [4].

Keskittämättömät sovellukset perustuvat Ethereum virtuaalikoneen (EVM, Ethereum Virtual Machine) kykyyn suorittaa Turing-täydellisiä ohjelmia lohkoketjussa. Näitä sovelluksia kutsutaan *älysopimuksiksi* (Smart Contract). Näkyviä sovelluskohteita ovat muun muassa joukkorahoittamiseen käytetyt haamuvaluutat sekä omistussuhteita kuvaavat tokenit.

4.5.1. Tavallinen tili

Ethereumissa käytetyt *tavalliset tilit* (EOA, Externally Owned Accounts) koostuvat Bitcoin-tilien tapaan *yksityisestä avaimesta*, *julkisesta avaimesta* sekä *osoitteesta*. *Yksityisavainta* käytetään transaktioiden allekirjoittamiseen. Transaktion allekirjoituksesta saadaan laskettua *julkinen avain*, josta voidaan johtaa *osoite* ja tarkistaa transaktion allekirjoituksen oikeellisuus. *Osoitteeseen* liittyvä varallisuus selvitetään laskemalla osoitekohtaiset transaktiot lohkoketjusta.

Ethereum transaktioiden allekirjoittamiseen käytetään Secp256k1-standardin avulla määriteltyä elliptistä käyrää [8][32].

- *Yksityisen avaimen* valinta tapahtuu valitsemalla positiivinen satunnaisluku joukosta $[1, secp256k1n]$, jossa $secp256k1n$ on ennaltamääritelty [32].
- *Julkinen avain* johdetaan yksityisavainta, elliptisellä käyrällä määriteltyä kertolaskua [14] sekä ennalta määriteltyä generoivaa alkiota [32] käyttäen.
- *Osoite* on yksityisen avaimen Keccak256-algoritmin avulla muodostetun hajautusarvon viimeiset 160 bittiä [8].

4.5.2. Älysopimustili

Tavallisten tilien lisäksi Ethereum esittelee älysopimuksiin liittyvät *sopimustilit* (contract accounts). Älysopimukset ovat ennen kaikkea automaattisia tilejä tai verkon autonomisia toimijoita, jotka suorittavat ja hyväksyvät transaktioita sopimustilin nykyiseen tilaan perustuen. Älysopimukseen liittyvä koodi tallennetaan lohkoketjun transaktioiden data-kenttään, josta seuraa että julkaistu älysopimus on jaettu kaikkien lohkoketjun osapuolten kanssa. Jaettua koodia suorittaa Ethereum virtuaalikone. Älysopimuksen kanssa vuorovaikuttava tili voi olla tavallinen tili tai älysopimus.

Älysopimuksen voidaan katsoa olevan tilakoneen kaltainen ohjelma, eli ohjelmakutsu voi vaikuttaa siihen, mitkä ohjelmakutsut ovat jatkossa mahdollisia. Esimerkiksi, älysopimus voi määrittää ylläpitäjien osoitteet sekä funktion, jonka avulla lisätään uusi ylläpitäjä. Esimerkissä ylläpitäjien joukkoon kuuluva tili voi tehdä ohjelmakutsun sisältävän transaktion, johon sisältyy parametrinä uuden ylläpitäjän osoite. Sellaisesta osoitteesta, joka ei kuulu sopimuksen määrittelemään ylläpitäjien joukkoon, lähetetty ohjelmakutsu ei esimerkin älysopimuksessa johda koskaan tilanmuutokseen.

4.5.3. Ethereum virtuaalikone

Ethereum virtuaalikone on osa Ethereum-lohkoketjusovellusta. Se suorittaa transaktioiden data-kentästä löytyviä ohjelmia tai ohjelmakutsuja ja varmistaa suorituksen jälkeisen uuden tilan oikeellisuuden. Ohjelmat ja ohjelmakutsut on tallennettu lohkoketjuun tavukoodina, joka määritellään Ethereumin käskykannassa [8].

Ethereum virtuaalikoneella suoritettavan ohjelman kuluttamat resurssit mitataan *kaasuna* (gas). Käskykannan eri operaatioille on määritelty kustannus (fee), jonka avulla hinnoitellaan jokaisen useita operaatioita sisältävän yksittäisen transaktion aiheuttama *kaasun kulutus*. Käyttäjä määrittelee transaktiolle *kaasuhinnan* (gas price) sekä *kaasurajan* (gas limit). Kaasuhinta määritellään Ethereum-valuutan avulla ja sillä on vaikutusta transaktion hyväksymisaikaan. Koska transaktion aiheuttama kaasun kulutus on joissakin tapauksissa etukäteen tuntematon, kaasurajalla ilmaistaan transaktion korkeinta sallittua kaasun kulutusta. Lopullinen transaktiokustannus saadaan laskemalla transaktiosta aiheutuvan kaasun kulutuksen ja transaktiossa määritellyn kaasuhinnan tulo.

4.5.4. Transaktio

Ethereumissa lohkot koostuvat transaktioista, jotka on serialisoitu RLP-enkoodausta (Recursive Length Prefix, rekursiivinen pituuden etuliite) käyttäen. Transaktio T koostuu allekirjoittamattomasta transaktiosta T_m sekä yksityisellä avaimella sk tehdystä todistuksesta s. Allekirjoittamaton transaktio muodostuu RLP-enkoodatuista parametreistä $T_m = RLP(nonce, kaasuhinta, kaasuraja, vastaanottaja, arvo, data)$. Todistus tuotetaan allekirjoittamalla enkoodattujen parametrien Keccak256-hajautusarvo $s = sign(keccak256(T_m), sk)$. Allekirjoitettu transaktio muodostetaan liittämällä todistus allekirjoittamattoman transaktion loppuun $T = RLP(T_m, s)$.

4.6. Kryptolompakot

Kryptolompakolla tarkoitetaan palvelua, ohjelmaa tai fyysistä laitetta, jonka avulla käyttäjä pääsee seuraamaan transaktioita tai vastaanottamaan ja kuluttamaan kryptovaluuttaa. Lompakon avulla käyttäjä hallitsee transaktioiden autentikointiin käytettyä yksityistä avainta.

Lompakko käyttää usein yhtä avainjuurta muodostamaan useita avaimia. Tällöin lompakon varmuuskopiointiin tarvitsee säilyttää ainoastaan avainjuuri. Usein avainjuuri muodostetaan satunnaisista sanoista koostuvan muistifraasin (mnemonic) avulla.

4.6.1. Pilvipohjainen lompakko

Palveluntarjoaja voi tarjota web-selaimella tai sovelluksella käytettäviä kryptolompakoita, joissa käyttäjän tunnistautumiseen käytetään perinteistä kolmannen osapuolen luottamusmallia. Kryptolompakon yksityinen avain voi olla ainoastaan palveluntarjoajan hallussa. Palveluntarjoaja voi mahdollistaa käyttäjälle helposti eri laitteilla saatavissa olevan lompakon ja esimerkiksi sähköpostiin perustuvan tilin palautuksen. Avainta säilytetään pilvessä ja se on altis palveluntarjoajan tai ulkopuolisten hyökkääjien hyväksikäytölle.

4.6.2. Lompakkosovellus

Lompakkosovellus toimii tietokoneella tai mobiilissa, ja sen avulla yksityisavaimia generoidaan ja säilytetään laitekohtaisesti. Sovellukset mahdollistavat usein avaimen varmuuskopioinnin muistifraasin avulla paperille, tai salasanalla kryptattuna erilliseen tiedostoon.

Lohkoketjusovellukset, jotka eivät ole erikoistuneet ainoastaan louhintaan, tarjoavat usein myös mahdollisuuden käyttää tilikohtaisia varoja. Lompakkosovellukset ovat yleisesti varsin turvallisia. Riskejä ovat avaimen varmuuskopioinnin laiminlyönti ja avaimia urkkivat haittaohjelmat.

4.6.3. Lompakkolaite

Käyttäjä voi hallita kryptovaluuttaa myös erityistehtävään suunnitelluiden lompakkolaitteiden avulla. Transaktion muodostamisen ajaksi laite yhdistetään esimerkiksi tietokoneeseen, jonka avulla allekirjoitettu transaktio lähetetään verkkoon. Transaktion hyväksymiseksi osa laitteista pyytää varmistukseksi myös pin-koodin.

Varmuuskopiointi lompakkolaitteissa perustuu muistifraasiin, joka voi olla kryptattu pin-koodilla. Lompakkolaitteet tarjoavat erittäin hyvän tietoturvan. Riskit liittyvät heikkoihin toteutuksiin ja käyttäjän lukittautumiseen ulos laitteelta.

5. SUOJATTU OHJELMAYMPÄRISTÖ

Ohjelmiston suojauksella tarkoitetaan ohjelman suorituksen ja sen käsittelemien tietojen turvaamista kaikilta ulkopuolisilta vaikuttajilta. Toisin kuin *normaalissa ympäristössä* suoritettava ohjelma, *suojattu ohjelma* ajetaan muusta järjestelmästä eristetyllä raudalla, jolloin ohjelman käsittelemät arkaluontoiset tiedot ovat salassa käyttäjiltä ja käyttöjärjestelmältä itseltään.

Kappaleessa käsitellään myös käyttöjärjestelmän rakennetta selventääksemme motivaatiota käyttää suojattua ohjelmaympäristöä sekä ymmärtääksemme suojaamattoman ympäristön uhkia.

5.1. Käyttöjärjestelmä ja suojaamattoman ohjelman haavoittuvuudet

Käyttöjärjestelmä on ohjelmisto, joka hallinnoi tietokoneen rauta- ja ohjelmistotason resursseja sekä tarjoaa palveluita ja vuorovaikutusrajapinnan järjestelmässä suoritettaville sovelluksille. Turvallisuuden näkökulmasta käyttöjärjestelmän tehtävänä on myös suojata resursseja, kuten prosessoria, käyttömuistia, massamuistia sekä ohjelmistoja käyttäjien tai haittaohjelmistojen pahansuovalta toiminnalta.

Tärkein menetelmä tiedon suojaukseen on segmentoida järjestelmä siten, että arkaluonteinen toiminta rajataan mahdollisimman pieneen ympäristöön. Etukäteen on kuitenkin vaikeaa määritellä mikä on arkaluonteista, normaalia tai mahdollisesti haitallista toimintaa. Puolestaan jokaisen toiminnan varmistaminen käyttäjältä on raskas ja hidas operaatio. Arkaluonteisia tietoja käsittelevien ohjelmien ja järjestelmien suunnittelussa joudutaankin usein tekemään kompromissi helpon käytettävyyden ja tiukan tietoturvan väliltä.

Erilaisiin tarkoituksiin spesifioidut ja konfiguroidut käyttöjärjestelmät toimivat erilaisella tietoturvapolitiikalla. Samalla kun arkipäiväiseen käyttöympäristöön tarkoitetut käyttöjärjestelmät ovat oletuksena sallivampia, niin kriittisessä ympäristössä resurssien käyttö halutaan määritellä tarkemmin. Käyttöjärjestelmissä käytetään tunkeutumisen havainnointisovelluksia sekä haittaohjelmaskannereita tunnistamaan ja eristämään haitallisia ohjelmia. Nämä sovellukset tarjoavat hyvin suojaa tunnettuja hyökkäyksiä ja haittaohjelmia vastaan, mutta jatkuvasti ilmestyy myös uusia tuntemattomia haittaohjelmia ja hyökkäyksiä.

5.2. Haittaohjelmat

Haitallisen sovelluksen on jossain tapauksessa mahdollista saastuttaa koko käyttöjärjestelmä, jonka jälkeen järjestelmän muistin ei voida ajatella olevan enää luottamuksellinen eikä eheä. Virtualisoinnin avulla haitallinen toimija voidaan pyrkiä eristämään virtuaaliympäristöön, on kuitenkin olemassa haavoittuvuuksia, joita hyödyntämällä virtuaaliympäristöstä on mahdollista saastuttaa virtuaaliympäristöä ylläpitävä ympäristö.

Käyttöjärjestelmä ja siinä suoritettavat ohjelmistot ovat erittäin monimutkaisia kokonaisuuksia, joten niiden kaikkea suorittamaa toimintaa on mahdotonta yleisesti todistaa virheettömäksi. Jotta jokin ohjelma voidaan suorittaa turvassa käyttöjärjestelmän sekä siinä suoritettavien ohjelmien virheiltä, ratkaisu on viedä suoritettava ohjelma käyttöjärjestelmän ulottumattomiin.

5.3. Suojatun ohjelmaympäristön rakenne

Suojattu ympäristö on muusta järjestelmästä eristetty, riippumattomia toimintoja tarjoava rautatason komponentin ohjelma. Yleinen tunnettu suojattu ympäristö on rautatason salausta tukeva massamuisti. *Suojattu ohjelmaympäristö* on sellainen suojattu ympäristö, johon käyttäjän on mahdollista kehittää itse suojattuja ohjelmia. Suojatut ohjelmaympäristöt on yleisesti toteutettu prosessorin laajennetulla käskykannalla, tai ovat erillisen prosessorin omaavia turvapiirejä (TPM, Trusted Platform Module).

5.4. Intel Software Guard Extension

Intel SGX (Software Guard Extension) on kokoelma prosessorikomentoja, jonka avulla luodaan *erillisalueella* (enclave) suoritettava *suojattu ohjelma* (trusted application). Erillisalueelle varataan käyttömuistista *suojattu muistialue*. Tämä muistialue on salattu prosessorin erillistä komentokantaa käyttäen ja siihen ei ole luku- tai kirjoitusoikeutta millään muulla prosessilla kuin erillisalueella suoritettavalla ohjelmalla.

Suojatun ohjelman lisäksi luodaan *normaaliohjelma* (untrusted application), jonka avulla erillisalue käynnistetään ja jolla ainoastaan voidaan tehdä ohjelmakutsuja suojattuun ohjelmaan. Normaaliohjelma on osa SGX-ohjelmaympäristöä ja vuorovaikuttaa suojatun ohjelman kanssa määritellyn rajapinnan kautta. Rajapinta kuvataan erillisohjelman kuvauskielellä (EDL, Enclave Definition Language) [33].

Lyhyesti esitettynä SGX:n uhkamalliksi on määritelty, että kaikki ulkopuoliset prosessit yrittävät urkkia toimintaa ja tietoja suojatulta ohjelmalta. SGX:n suunnittelun periaatteita ovat [34].

1. Mahdollistaa kehittäjille arkaluonteisen tiedon suojaaminen korkeammilla prioriteeteilla toimivien haittaohjelmien luvattomalta käytöltä ja muokkaukselta.

2. Varmistaa sovelluksien arkaluontoisen koodin ja tietojen luottamuksellisuus ja eheys häiritsemättä oikeutettujen laiteohjelmistojen kykyä ajastaa ja hallita laitteen resurssien käyttöä.

3. Mahdollistaa käyttäjän hallita järjestelmäänsä sekä asentaa ja poistaa sovelluksia ja palveluita valitsemallaan tavalla.

4. Sallia alustan analysoida sovelluksen suojattu koodi ja allekirjoittaa todistus, että koodi on alustettu asianmukaisesti luotetussa ympäristössä.

5. Mahdollistaa luotettujen sovellusten kehittäminen tunnettujen työkalujen ja menetelmien avulla.

6. Mahdollistaa suorituskyvyn skaalautuminen prosessorin ominaisuuksien mukaan.

7. Mahdollistaa luotettujen sovellusten ja päivitysten toimittaminen kehittäjien valitsemien jakelukanavien avulla.

8. Antaa sovellusten määritellä suojatut alueet koodista ja tietueista, jotka säilyttävät luottamuksellisuuden, vaikka hyökkääjällä olisi fyysinen kontrolli alustaan ja kyky suorittaa suoria hyökkäyksiä muistia vastaan.

6. SUUNNITELMAN KUVAUS

Projektin lähtökohtana on yhdistää lompakkolaitteiden, lompakkosovellusten ja pilvipohjaisten lompakoiden hyviä ominaisuuksia erityisesti tiedon luottamuksellisuutta, eheyttä ja saatavuutta silmällä pitäen. Suunniteltava sovellus tarjoaa turvallisen säilytysympäristön lohkoketjussa hyödynnettävälle kryptografiselle avaimelle käyttämällä avainta selkokielisenä ainoastaan suojatussa ohjelmaympäristössä sekä mahdollistamalla avaimen monivaiheisen varmuuskopioinnin ja palauttamisen.

Vaikka Shamirin salaisuuden jaon menetelmää käytetään yleisesti avaimen jakamiseen toisistaan riippumattomiin osiin, ja suojattua ympäristöä on käytetty muun muassa lompakkosovellusympäristöinä; työssä esitellään tiettävästi ensimmäinen kuvaus lohkoketjuissa käytettävän kryptografisen avaimen suojatusta hallintaympäristöstä, joka mahdollistaa avaimen varmuuskopioinnin useisiin toisistaan riippumattomiin osatietoihin.

Kuvauksessa suunniteltavaa sovellusta voi käyttää joko ihminen tai tietokone, esimerkiksi lohkoketjuun yhteydessä oleva IoT-laite.

6.1. Uhkamalli

Mallin edellytetään ratkaisevan seuraavat tilanteet:

1. *Laitteen rikkoutuminen.* Laitteen rikkoontuessa avaimen täytyy olla palautettavissa.

2. *Pilvipalvelussa säilytetyn datan vuotaminen.* Vaikka pilvipalveluntarjoaja altistuisi datavuodolle, kenelläkään ei saa olla mahdollisuutta käyttää avainta.

3. *Tiedonkalastelu.* Avainta ei näytetä selkokielisenä käyttäjälle.

4. *Laitteen joutuminen haitalliselle toimijalle.* Avain ei saa olla hyödynnettävissä, vaikka laite joutuisi vääriin käsiin.

5. *Mahdollisuus asentaa haittaohjelmia.* Vaikka laitteen käyttöjärjestelmään on mahdollisuus asentaa haittaohjelmia, kenelläkään ei saa olla mahdollisuutta hyödyntää avainta.

6. *Haitallisen toimijan pääsy laitteen käyttöjärjestelmään.* Avain ei saa olla hyödynnettävissä, vaikka hyökkääjällä on pääsy laitteen käyttöjärjestelmään.

6.2. Suojattavat toiminnot

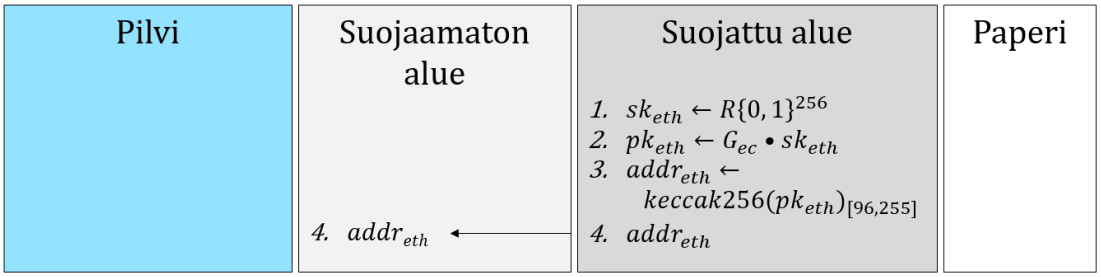
Suunnitelmassa kuvataan avaimen varmuuskopiointi kahteen toisistaan riippumattomaan osatietoon, jotka tallennetaan pilvipalvelimelle ja paperille. Turvallisuuden kannalta kaikki avaimen käsittelyyn liittyvät toiminnot

ovat kriittisiä. Näitä operaatioita ovat avaimen generointi, transaktioiden allekirjoittaminen, osatietojen varmuuskopiointi sekä osatietojen palautus. Kaikki varsinaiseen avaimeen liittyvät operaatiot sekä avaimen säilyttäminen tapahtuvat suojatussa ohjelmaympäristössä.

6.2.1. Avaimen generointi

Ethereumissa käytettävän avainparin generointi tapahtuu suojatussa ohjelmaympäristössä, Kuva 9. Tilikohtainen osoite jaetaan suojaamattomaan ympäristöön, jotta transaktioiden muodostaminen on mahdollista.

1. Suojatussa ympäristössä generoidaan satunnaisluku, jota käytetään yksityisenä avaimena sk_{eth}.
2. Yksityisestä avaimesta sk_{eth} johdetaan julkinen avain pk_{eth}.
3. Lasketaan Ethereum-osoite $addr_{eth} = keccak256(pk_{eth})_{[96,255]}$, 160-bittiä.
4. Julkaistaan osoite suojaamattomaan ympäristöön.

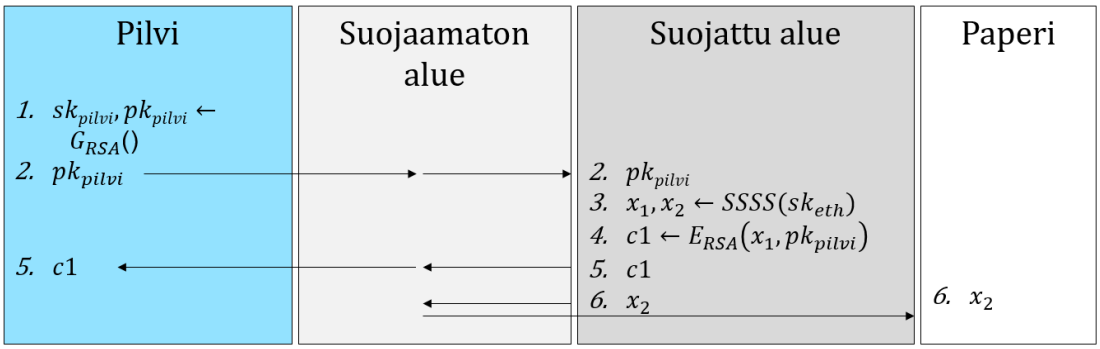


Kuva 9. Avaimen generointi

6.2.2. Avaimen varmuuskopiointi

Avaimen varmuuskopiointi tehdään kahden toisistaan riippumattoman osatiedon avulla. Viestit pilvipalveluntarjoajalle ja fyysiseen kopioon lähetetään suojaamattoman ohjelman yli, jonka takia viestit voivat altistua suojaamattoman puolen urkinnalle. Avaimen osatietojen turvaamiseksi etäpalvelimelle lähetettävät viestit salataan epäsymmetristä salausmenetelmää käyttäen, Kuva 10.

1. Pilvipalvelu generoi RSA-avainparin (sk_{pilvi}, pk_{pilvi}).
2. Pilvipalvelimen julkinen avain pk_{pilvi} siirretään suojatulle alueelle.
3. Suojatussa ympäristössä Ethereum yksityinen avain sk_{eth} jaetaan Shamirin salaisuuden jaon avulla osatietoihin x_1 ja x_2.
4. Osatieto x_1 salataan käyttäen pilven julkista salausavainta pk_{pilvi}.
5. Salattu osatieto c_1 siirretään pilvipalvelimelle.
6. Selkokielinen osatieto x_2 tulostetaan paperille.

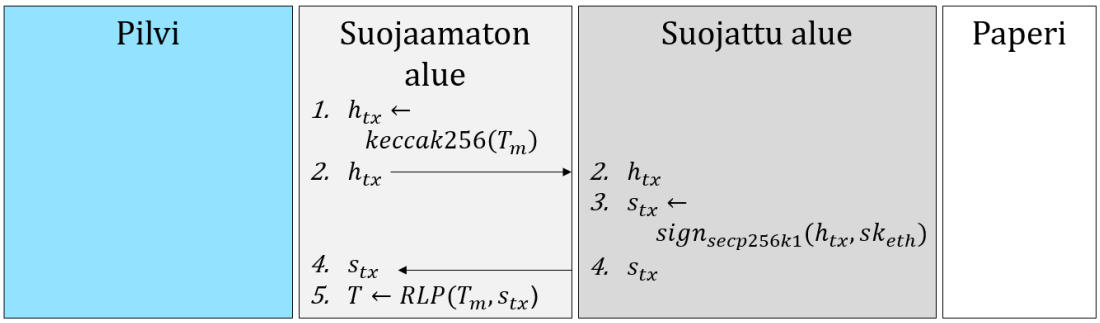


Kuva 10. Avaimen varmuuskopiointi

6.2.3. Transaktion allekirjoittaminen

Transaktion muodostaminen ja lähettäminen tapahtuu suojaamattomalla puolella, koska prosessi vaatii vuorovaikutusta käyttäjän sekä lohkoketjuverkon kanssa. Transaktion allekirjoittaminen tapahtuu suojatulla puolella, Kuva 11.

1. Suojaamattomassa ympäristössä generoidaan allekirjoittamaton transaktio T_m ja lasketaan sen tiiviste h_{tx}.
2. Siirretään tiiviste suojatulle alueelle.
3. Muodostetaan todistus s_{tx} allekirjoittamalla tiiviste käyttäen yksityistä avainta sk_{eth} Ethereumin noudattamalla elliptisellä käyrällä $secp256k1$.
4. Siirretään todistus suojaamattomaan suoritusympäristöön.
5. Muodostetaan allekirjoitettu transaktio T liittämällä todistus s_{tx} allekirjoittamattoman transaktion T_m loppuun käyttäen RLP-enkoodausta $T = RLP(T_m, s_{tx})$.

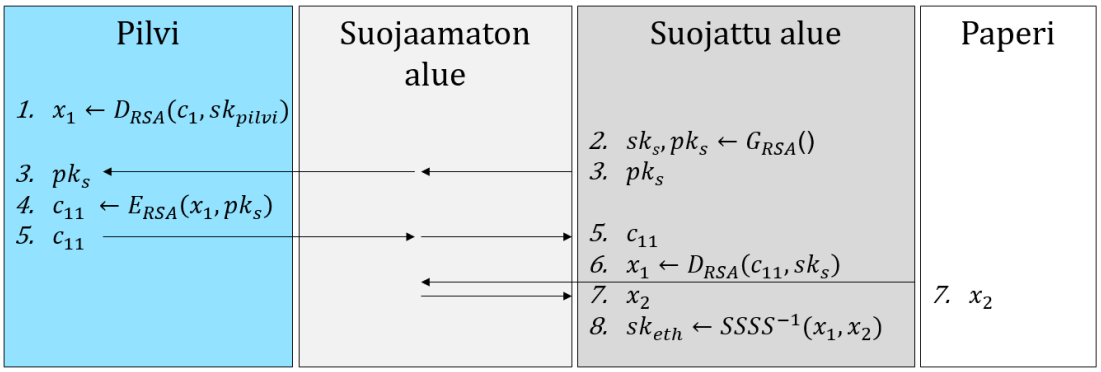


Kuva 11. Transaktion allekirjoittaminen

6.2.4. Avaimen palauttaminen

Avaimen palauttamiseksi osatiedot tuodaan suojaamattoman ympäristön yli suojatulle puolelle. Etäpalvelimelta lähetettävät osatiedot salataan epäsymmetristä kryptografiaa käyttäen, Kuva 12.

1. Pilvipalvelin purkaa salatun osatiedon c_1 käyttäen omaa yksityistä avainta sk_{pilvi} ja muodostaen salaamattoman osatiedon x_1.
2. Suojatussa ympäristössä generoidaan RSA-avainpari (sk_s, pk_s).
3. Suojatun ympäristön julkinen avain pk_s siirretään pilvipalvelimelle.
4. Pilvipalvelin salaa osatiedon x_1 käyttäen suojatun ympäristön julkista avainta pk_s, muodostaen salatun osatiedon c_{11}.
5. Salattu osatieto siirretään suojattuun ympäristöön.
6. Salattu osatieto puretaan käyttäen suojatun ympäristön yksityistä avainta sk_s, muodostaen salaamattoman osatiedon x_1.
7. Paperilla oleva osatieto luetaan ja palautetaan suojattuun ympäristöön.
8. Käytetään Shamirin salaisuuden jaon palautusfunktiota paljastamaan Ethereum yksityisavain sk_{eth}.



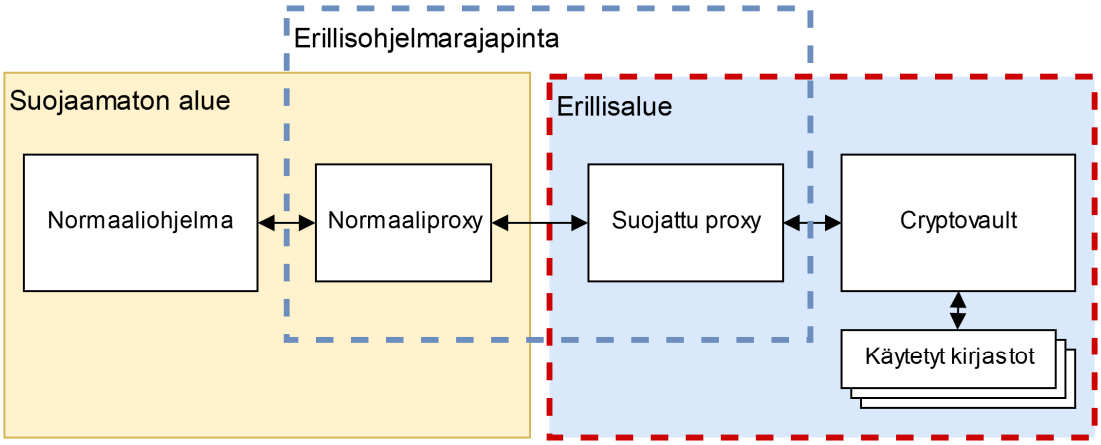
Kuva 12. Avaimen palauttaminen

7. CRYPTOVAULT

Cryptovault on sovellus, joka pyrkii tarjoamaan varmuuskopioitavan luotettavan avainsäilön pohjautuen edellisen kappaleen kuvaukseen sekä Intel SGX suojattuun ohjelmaympäristöön

7.1. Ohjelman rakenne

Ohjelma koostuu SGX:lle tyypillisesti suojaamattomalla puolella olevasta *normaaliohjelmasta* sekä suojatun ohjelmaympäristön erillisalueella toimivasta *suojatusta ohjelmasta*. Lisäksi ohjelmaan kuuluu erillisalueen konfiguraation sisältävä tiedosto sekä normaaliohjelman ja suojatun ohjelman välisen erillisohjelmarajapinnan määrittelevä EDL-tiedosto. Ohjelma on toteutettu C++-kielellä ja se käyttää myös C-kielisiä kirjastoja. Sovelluksen rakenne on esitetty Kuvassa 13.



Kuva 13. Sovelluksen rakenne

7.2. Suojattu ohjelma

Suojatun ohjelman toteutuksessa on käytetty Intel SGX ohjelmakehitykseen vaadittavia ajureita ja kirjastoja [35]. Ohjelma hyödyntää elliptisten käyrien allekirjoittamiseen käytettävää Secp256k1-kirjastoa [36] sekä Shamirin salaisuuden jakamisen toteuttavaa kirjastoa [37].

Ethereumissa käytettävä avainpari generoidaan suojatussa ohjelmassa, joka tallettaa yksityisen avaimen salattuun tiedostoon käyttäen SGX-tason salausta (sealing). Yksityinen avain on mahdollista jakaa osatietoihin sekä palauttaa osatiedoista käyttäen Shamirin menetelmää.

Suojatun ohjelman avulla voidaan generoida 2048-bittisiä RSA-avaimia sekä salata ja purkaa viestejä käyttäen RSA-salausmenetelmää. Lisäksi suojatussa

ympäristössä on mahdollista allekirjoittaa viestejä käyttäen elliptisen käyrän menetelmää Secp256k1:ssä määriteltyjen parametrien kanssa.

7.3. Normaaliohjelma

Ohjelma koostuu suojatun ympäristön hallitsemiseen vaadittavista operaatioista ja komentoparametritulkista. Lisäksi ohjelma sisältää tiedoston tuonti- ja vientifunktiot, joiden avulla esimerkiksi transaktioiden hajautusarvot ja näiden allekirjoitukset saadaan siirrettyä normaaliohjelman ja suojatun ympäristön välillä. Normaaliohjelma tulkkaa käyttäjän syöttämät komentoriviparametrit, käynnistää erillisalueen, suorittaa tulkatut käskyt ja sulkee erillisalueen.

7.4. Käyttökohteet

Sovellusta voidaan käyttää järjestelmässä, jossa prosessori ja emolevy tukevat Intel SGX ympäristöä. Yritys- ja kuluttajatietokoneiden lisäksi Intel SGX sovelluksia tukevia ympäristöjä on tarjolla myös pilvipalveluissa [38]. Vaikka Intel SGX on heikosti saatavilla etenkin mobiililaitteille, on olemassa myös muiden valmistajien toteuttamia suojattuja ohjelmaympäristöjä, johon samankaltainen sovellus voidaan toteuttaa.

Mallissa esitetyn pilvipalvelun tehtävät voidaan toteuttaa käyttäen Cryptovault-ohjelmaa. Pilvipalvelun toimintoja ovat osatietojen säilyttäminen, RSA-avaimen generoiminen ja viestien salaaminen ja purkaminen.

7.5. Toiminta

Cryptovault toteuttaa kaikki mallissa kuvailtuun varmuuskopiointiin vaadittavat operaatiot. Totetutusta ei kuitenkaan pystytä käyttämään varsinaisena avainsäilönä johtuen siitä, että mallissa kuvailtujen operaatioiden ketjuttaminen ei onnistu nykyisen toteutuksen turvin. Suojattuun ympäristöön määritettävän rajapinnan kannalta on ratkaisevaa se, miten sovellusta halutaan käyttää. Erillisalueen ohjelmarajapinta täytyy suunnitella siten, että hyökkäyspinta suojattua ohjelmaa vastaan on mahdollisimman pieni. Seuraavassa kappaleessa käydään läpi erilaisia hyökkäyksiä suojattua ohjelmaa vastaan.

8. POHDINTA

Järjestelmän tuoma lisäturva perustuu kriittisen tiedon jakamiseen riippumattomissa osissa usealle palveluntarjoajalle. Kappaleessa käydään läpi uhkakuvat esitettävien hyökkäysten kautta, ja pohditaan mitä seurauksia uhkakuvien toteutuessa voi ilmetä. Lisäksi kappaleessa ehdotetaan lisäturvaa tuovia ominaisuuksia hyökkäyskanavia vastaan.

8.1. Seuraukset

Suunnitelmassa kuvattujen uhkakuvien toteutuessa seuraukset voidaan jakaa alle listattuihin tapahtumiin.

1. *Avaimen haltuunsaanti.* Kokonaisen avaimen haltuunsaamiseksi hyökkääjän täytyy pystyä pääsemään käsiksi joko *avaimeen* tai *molempiin osatietoihin.*
2. *Mahdollisuus käyttää avainta.* Jotta hyökkääjä voi käyttää avainta, hänellä tulee olla se hallussaan tai hänen tulee pystyä ohjaamaan avaimen sisältävää suojattua ohjelmaa.
3. *Osatiedon haltuunsaanti.* Avaimen haltuunsaaminen helpottuu, jos hyökkääjä saa haltuunsa yhdenkään osatiedon.
4. *Tilin käytön estäminen.* Hyökkääjä voi onnistua estämään sovelluksen käytön. Jos varmuuskopion osatiedot ovat tallessa, käyttäjä voi palauttaa avaimen.
5. *Osatiedon hukkaaminen.* Liian monen osatiedon hukkaaminen estää varmuuskopion palauttamisen.

8.2. Turvallisuusanalyysi

Sovelluksen tietoturvallisuusominaisuudet analysoidaan uhkamallin ja kuvitteellisen hyökkääjän avulla. Hyökkääjän oletetaan kykenevän kuuntelemaan ja muokkaamaan jokaista suojatun ympäristön ulkopuolella tapahtuvaa operaatioita, ellei sitä ole erikseen tapauskohtaisesti rajattu. Jokaisessa kappaleessa esitellään hyökkäysmenetelmä, miten hyökkäykseltä pyritään suojautumaan, mitä seurauksia hyökkäyksestä voi aiheutua sekä voidaanko hyökkäykseltä suojautua muilla toimenpiteillä.

Aktiivinen välimieshyökkäys

Varmuuskopioinnin tekemisen sekä varmuuskopion palauttamisen aikana suojaamattomalla puolella on mahdollisuus tehdä aktiivinen välimieshyökkäys, jonka avulla saadaan luettua pilveen lähetettävät osatiedot.

Nykyisessä toteutuksessa sekä pilven että suojatun ympäristön julkinen avain tallennetaan salaamattomaan tiedostoon. Tästä seuraa, että haittaohjelman, jolla on kirjoitusoikeus kansioon, on mahdollista generoida oma RSA-avainpari ja muuttaa, esimerkiksi varmuuskopioinnin aikana, pilven julkinen avain omaan julkiseen avaimeen. Hyökkäys johtaa *osatiedon haltuunsaamiseen*. Hyökkäys on mahdollista ainoastaan varmuuskopioinnin tai palautuksen aikana.

Parempi ratkaisu olisi toteuttaa suora yhteys normaaliohjelman ja etäpalvelimen välille varmuuskopioinnin ja palauttamisen ajaksi, jolloin välimieshyökkäys olisi mahdollista enää verkossa tai etäpalvelimen suorittavalla ohjelmalla. Turvallinen ratkaisu välimieshyökkäyksen estämiseksi olisi symmetrinen salaus etäpalvelimen ja suojatun ohjelman välillä. Huomioitavaa on, että avaimen jakelusta muodostuisi mahdollisesti alkuperäistä avaimen säilyttämistä vastaava haaste. Käytettävyyttä silmälläpitäen muita ratkaisuja välimieshyökkäyksen estämiseen ovat etäpalvelimelle ennaltasovittava sertifikaatti tai viestien tarkka ajoitusanalyysi [39].

Laitteen fyysinen haltuunsaanti

Hyökkääjällä, joka saa laitteen fyysisesti haltuunsa pääsemättä käsiksi käyttöjärjestelmään, ei ole mahdollista urkkia avainta laitteesta perustuen SGX:n turvallisuuteen. Jos käyttäjä menettää laitteensa hallinnan, hyökkääjä onnistuu *tilin käytön estämisessä*. Käyttäjä voi palauttaa avaimen osatiedoista.

Haitallisen toimijan pääsy laitteen käyttöjärjestelmään

Tilanteessa, jossa hyökkääjä pääsee laitteen käyttöjärjestelmään ilman oikeutta suorittaa normaaliohjelmaa, ei seuraa välittömiä uhkia sillä oletuksella, että käyttäjä on tehnyt avaimesta varmuuskopion ja osatietoja ei säilytetä järjestelmässä. Jos hyökkäystä ei huomata, hyökkääjällä voi olla mahdollisuus *ohjelman hyväksikäyttöön*.

Jos hyökkääjä saa lukuoikeuden muistialueelle, johon paperille tulostettava varmuuskopio väliaikaisesti ladataan, seuraa *osatiedon haltuunsaanti*. Hyökkääjän saadessa luku- ja kirjoitusoikeudet kansioon, johon julkiset avaimet tallennetaan käyttäjän käynnistämän varmuuskopioinnin ja palautuksen aikana, hyökkääjällä on mahdollisuus tehdä *aktiivinen välimieshyökkäys*. Jos hyökkääjä puolestaan saa suojatun ohjelman käynnistämiseen vaadittavat oikeudet, hänellä on mahdollisuus tehdä varmuuskopio avaimesta, joka johtaa *avaimen haltuunsaantiin*. Vaihtoehtoisesti hyökkääjällä on *mahdollisuus käyttää avainta* sekä *estää tilin käyttäminen*.

Paremman turvallisuuden saavuttamiseksi, käyttäjältä voidaan vaatia salasana tai pin-koodi erillisalueen käynnistämiseksi. Tästä seuraisi suurempi avaimenkadottamisriski.

Ohjelman hyväksikäyttö

Hyökkäys transaktion muodostamiseen käytettyä ohjelmaa vastaan voi mahdollistaa muun muassa maksunsaajan osoitteen muuttamisen hyökkääjän omaksi osoitteeksi eli hyökkääjällä on *mahdollisuus käyttää avainta*. Esimerkiksi on olemassa hyökkäyksiä, joissa leikepöydälle kopioitu osoite korvataan hyökkääjän samankaltaisella osoitteella [40].

Paremman turvallisuuden saavuttamiseksi transaktio tulisi siirtää kokonaisuudessaan suojattuun ympäristöön tai transaktion muodostus tulisi tehdä suojatussa ympäristössä käyttäjän antamilla parametreillä. Tällöin käyttäjältä voidaan kysyä vahvistus, onko transaktio tarkoitetun kaltainen.

Pilvipalveluun tunnistautumisen vaarantuminen

Tilanteessa, jossa hyökkääjä onnistuu tunnistautumaan etäpalvelimelle varmuuskopion tehneen käyttäjän nimissä, voi hyökkääjä palauttaa varmuuskopion johtaen *osatiedon haltuunsaantiin*. Lisäksi on mahdollisuus, että hyökkääjä poistaa osatiedon tai vaihtaa tunnistautumiseen vaadittavan salasanan johtaen *osatiedon hukkaamiseen*.

Hyökkäyksen vaikutusten minimoimiseksi ehdotetaan vähintään yhtä paperille sekä kolmea etäpalvelimelle lähetettävää osatietoa, joista mitkä tahansa kolme riittää avaimen palauttamiseksi. Tällöin yhden paljastuneen osatiedon vaikutus ei ole niin suuri kuin tilanteessa, jossa vaaditaan kaksi osatietoa avaimen palauttamiseen. Myöskään yhden osatiedon hukkaaminen ei kuvatussa tilanteessa estä avaimen palauttamista.

Aikalukon avulla osatiedon palauttamiseen voidaan tehdä rajoite. Menetelmän varmuuskopiovaiheessa Ethereumin julkinen avain saatetaan pilvipalvelun tietoon. Osatiedon palauttamispyynnön jälkeen pilvipalvelu odottaa aina ennalta määrätyn ajan ennen osatiedon lähettämistä. Jos odotusaikana pilvipalveluun lähetetään yksityisavaimella allekirjoitettu viesti palautuksen estämiseksi, niin palauttaminen perutaan. Menetelmä vaatii pilvipalvelun palautuspyyntöjen aktiivista monitorointia.

Heikko kryptografia

Suojatussa ohjelmassa ja etäpalvelimella tehtävä RSA-salaus on toteutettu 2048-bittistä salausavainta käyttäen, joka on myös suositeltu RSA-avaimen pituus [41]. Tilanteessa, jossa hyökkääjällä olisi käytössään tehokas menetelmä purkaa 2048-bittisellä RSA-avaimella salattu viesti, hänellä on mahdollisuus *osatiedon haltuunsaantiin*. Lisäturvaa voi saavuttaa RSA-avainpituutta kasvattamalla, tai käyttämällä ECC-salausmenetelmää tarkoituksenmukaisella avainpituudella. Vertailemalla RSA- ja ECC-menetelmiä on ehdotettu, että 224-bittisellä avaimella tehty ECC-salaus vastaa turvallisuudeltaan 2048-bittisellä avaimella tehtyä RSA-salausta [16].

Tarkastelun ulkopuolelle jäävät Ethereumin valitsemat kryptoprimitiivit, primitiivit joiden avulla SGX on toteutettu sekä käytettyjen kirjastojen ennalta tuntemattomat heikkoudet.

Paperin vahingollinen säilyttäminen

Jos osatiedosta tehdään fyysinen kopio paperille, paperin tulostamisen aikana osatieto on järjestelmän ja tulostimen muistissa. Muistialueen tai paperin altistuminen hyökkääjän luettavaksi johtaa *osatiedon haltuunsaantiin* ja mahdollisesti *osatiedon hukkaamiseen*.

Tiedonkalastelu

Ohjelma ei paljasta kryptografista avainta edes käyttäjälle. Ainut menetelmä avaimen selvittämiseksi on palauttaa avain osatiedoista. Osatietoja ei ole varsinaisesti salattu käyttäjältä, joten tiedonkalastelun riski on olemassa.

Laitteen rikkoontuminen

Laitteen rikkoontuminen johtaa *tilin käytön estymiseen*. Käyttäjä voi palauttaa avaimen osatiedoista.

Uhkamallin ulkopuoliset tekijät

Yleinen heikkous lompakkosovelluksissa ja -laitteissa on satunnaislukugeneraattori. Tällöin yksityinen avain saatetaan löytää erilaisia avaimia kokeilemalla (brute force). Tällä hetkellä sovellus käyttää SGX:n satunnaislukugeneraattoria. Vaihtoehtoinen ratkaisu olisi koostaa avain eri lähteistä kerätyistä satunnaisluvuista.

Riskiarvioinnin ulkopuolelle on jätetty suojattua ympäristöä koskevat haavoittuvuudet. Näitä ovat muun muassa vuonna 2017 löydetyt SGX:n liittyvät välimuisti haavoittuvuudet [42]. Lisäksi vuonna 2018 modernista prosessiarkkitehtuurista on löydetty spekulatiiviseen suorittamiseen liittyviä suunnitteluvirheitä, jotka paljastavat myös Intel SGX:ssä käsiteltäviä tietoja [43].

8.3. Mallin ja sovelluksen vahvuudet

Suojattu ohjelmaympäristö on turvallinen avainsäilö niin kauan, kun avain generoidaan ja säilytetään siellä. Uhkakuvien perusteella, osatietojen turvallinen varmuuskopioiminen vaatii vähintään yhteen fyysiseen säilöön sekä kolmeen eri etäpalveluun tallennettavia osatietoja, joista avaimen palauttamiseen vaaditaan mitkä tahansa kolme osatietoa. Lisäksi käyttäjän kannattaa tehdä ennalta sovitut

sertifikaatit pilvipalveluiden kanssa. Näillä edellytyksillä edellä kuvattu malli mahdollistaa turvallisen tavan säilyttää lohkoketjuavainta.

Pilvipohjaiseen lompakkoon verrattuna kuvatun menetelmän vahvuus on, ettei luottamus rakennu ainoastaan yhden palveluntarjoajan varaan. Lompakkosovellukseen verrattuna esitelty menetelmä kuvaa turvallisemman tallennusympäristön avaimelle. Lompakkolaitteiden heikkoutena työssä kuvattua mallia vastaan voidaan nähdä usein selkokielisenä muistifraasina tehtävä varmuuskopio.

8.4. Mallin ja sovelluksen heikkoudet

Uhkakuvien perusteella lohkoketjuavaimen käyttäminen turvallisesti järjestelmässä, joka on hyökkääjän hallussa, ei näytä mahdolliselta. Toteutettu sovellus ei selviydy uhkamallin kuvaamista tilanteista, koska useissa tilanteissa turvallisuus pohjautuu enää käyttöjärjestelmän turvallisuuteen. Työssä aiemmin esitettyjen menetelmien avulla sovelluksen turvallisuutta pystytään parantamaan.

Koska vuorovaikutus suojatun ympäristön kanssa pohjautuu myös käyttöjärjestelmän turvallisuuteen, kannattaa huomioida myös seuraavat toimenpiteet:

- Paperille tulostettava osatieto tulee tulostamisen jälkeen poistaa turvallisesti muistista
- Käyttöjärjestelmä tulee suojata vahvalla salasanalla
- Kansion luku- ja kirjoitusoikeudet tulee rajata vierastunnuksilta

8.5. Lohkoketjuteknologian tulevaisuus

Vaikka lohkoketjuteknologia esittelee mallin keskittämättömälle luottamukselle, suhtautuminen lohkoketjuihin on usein epäröivää. On esitetty kysymys, jos lohkoketjuteknologia tarjoaa ylivertaista luottamusta verrattuna nykyiseen kolmannen osapuolen luottamusmalliin, miksi lohkoketjuilla on perinteisiä palveluita suurempia ongelmia valuuttavarkauksien ja -huijauksien suhteen [44].

Yksi syy voi löytyä lohkoketjuteknologian ominaisuuksista, etenkin toisilleen tuntemattomien ihmisten vuorovaikutuksesta. Emme ole koskaan siirtäneet rahaa yhtä anonyymisti ja säännöstelemättömästi. Etenkin internetin kehittymisen myötä, toisilleen tuntemattomien ihmisten viestintää on tutkittu, ja havaittu anonyymiuden madaltavan kynnystä negatiivisempaan viestintään [45]. Lisäksi on todettu, että anonyymius poistaa meille tyypillistä käyttäytymiseen vaikuttavaa sosiaalista painetta [46].

Meille ehkä lähtökohtaisesti riittäisi, että kolmannen osapuolen luottamusmalli toimii tällä hetkellä. Olemme kuitenkin huonoja varautumaan poikkeustilanteisiin. Lohkoketjujen voidaan katsoa olevan puolueettomia, ja täten tarjoavan turvaa myös tilanteissa, joissa yksittäiset toimijat eivät halua tehdä yhteistyötä muiden kanssa. Toisaalta, lohkoketjut ovat erittäin uusi teknologia.

Emme mitenkään voi saada etukäteen selville, onko lohkoketjuteknologia ratkaisu tulevaisuuden tuntemattomiin ongelmiin.

Käyttäjien ja palveluntarjoajien lisäksi luottamusta tarvitaan myös palveluntarjoajien kesken. Ehkä tulevaisuudessa loppukäyttäjät tunnistautuvat palveluntarjoajille edelleen perinteisin menetelmin ja lohkoketjut tuovatkin luottamusta palveluntarjoajien välille.

Käyttötarkoituksesta riippumatta työssä esitelty malli on yksi tietoturvallinen tapa säilöä lohkoketjuavain.

9. YHTEENVETO

Työssä esiteltiin sekä malli että toteutus lohkoketjuavaimen säilyttämiseen suojatussa ohjelmaympäristössä. Mallin erityispiirteitä olivat kryptografisen avaimen turvallinen varmuuskopiointi Shamirin salaisuuden jaon avulla ja suojattuun ympäristöön toteutettu avainsäilö. Toteutuksessa suojattuna ohjelmaympäristönä hyödynnettiin Intel SGX käskykantaa.

Suunnitelman ja toteutuksen ratkaisut käytiin läpi objektiivisen uhkamallin kautta. Vaikka turvallisuutta kunnioittava avaimen varmuuskopiointi onnistuttiin saavuttamaan, kaikkia riskejä ei voida poissulkea nykyisen toteutustavan turvin.

Nykyisen toteutuksen turvallisuutta pystytään parantamaan pohdinnassa esitetyillä toimenpiteillä. Käsittelemättä jää, voidaanko mahdolliset ohjelmaympäristön tekniset haavoittuvuudet kartoittaa.

10. LÄHTEET

[1] Schneier B. (2016) Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World. W. W. Norton, 92-98 p.

[2] Demirguc-Kunt A., Klapper L., Singer D. & Ansar S. (2018) The Global Findex Database 2017: Measuring Financial Inclusion and the Fintech Revolution. World Bank Publications, 1-10 p.

[3] McCandless D., Evans T., Barton P., Tomasevic S. & Geere D. (2019), World's biggest data breaches and hacks (luettu 30.4.2019). https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/.

[4] Nakamoto S. (2008), Bitcoin: A peer-to-peer electronic cash system (luettu 6.5.2019). https://bitcoin.org/bitcoin.pdf.

[5] Kartalopoulos S.V. (2006) A primer on cryptography in communications. IEEE Communications Magazine 44, pp. 146–151.

[6] Menezes A.J., Van Oorschot P.C. & Vanstone S.A. (1996) Handbook of applied cryptography. 323–330 p.

[7] Kelsey J. & Schneier B. (2005) Second preimages on n-bit hash functions for much less than 2nwork. In: R. Cramer (ed.) Advances in Cryptology – EUROCRYPT 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 474–490.

[8] Wood G. (2014), Ethereum: A secure decentralised generalised transaction ledger (luettu 6.5.2019). https://ethereum.github.io/yellowpaper/paper.pdf.

[9] Bertoni G., Daemen J., Peeters M. & Van Assche G. (2011), Cryptographic sponge functions (luettu 10.5.2019). https://keccak.team/files/CSF-0.1.pdf.

[10] Dwork C. & Naor M. (1992) Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference, Springer, pp. 139–147.

[11] Back A. (2002), Hashcash - a denial of service counter-measure (luettu 7.5.2019). http://www.hashcash.org/papers/hashcash.pdf.

[12] Merkle R.C. (1988) A digital signature based on a conventional encryption function. In: C. Pomerance (ed.) Advances in Cryptology — CRYPTO '87, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 369–378.

[13] Rivest R.L., Shamir A. & Adleman L. (1978) A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM .

[14] Miller V.S. (1986) Use of elliptic curves in cryptography. In: H.C. Williams (ed.) Advances in Cryptology — CRYPTO '85 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 417–426.

[15] Linna J. (2005) Elliptisiin käyriin perustuvat kryptosysteemit. Diplomityö, Tampereen teknillinen yliopisto, Tietotekniikan osasto.

[16] Gura N., Patel A., Wander A., Eberle H. & Shantz S.C. (2004) Comparing elliptic curve cryptography and rsa on 8-bit cpus. In: M. Joye & J.J. Quisquater (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 119–132.

[17] Miller F. (1882) Telegraphic code to insure privacy and secrecy in the transmission of telegrams. CM Cornwell.

[18] Bellovin S.M. (2011) Frank miller: Inventor of the one-time pad. Cryptologia 35, pp. 203–222.

[19] Shamir A. (1979) How to share a secret. Commun. ACM 22, pp. 612–613. URL: http://doi.acm.org/10.1145/359168.359176.

[20] Waring E. (1779) Vii. problems concerning interpolations. Philosophical transactions of the royal society of London , pp. 59–67.

[21] Palatinus M., Rusnak P., Voisine A. & Bowe S. (2013), Mnemonic code for generating deterministic keys (luettu 22.5.2019). https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki.

[22] Kaliski B. (2000) Password-based cryptography specification version 2.0. RFC 2898 https://tools.ietf.org/html/rfc2898.

[23] Raval S. (2016) Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly Media, 1–14 p.

[24] Akkoyunlu E.A., Ekanadham K. & Huber R.V. (1975) Some constraints and tradeoffs in the design of network communications. In: ACM SIGOPS Operating Systems Review, vol. 9, ACM, vol. 9, pp. 67–74.

[25] Lamport L., Shostak R. & Pease M. (1982) The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) 4, pp. 382–401.

[26] Haber S. & Stornetta W.S. (1991) How to time-stamp a digital document. Journal of Cryptology 3, pp. 99–111. URL: https://doi.org/10.1007/BF00196791.

[27] Bayer D., Haber S. & Scott Stornetta W. (1992) Improving the efficiency and reliability of digital time-stamping .

[28] Viitala J. (2016), Mikä on lohkoketju? (luettu 9.5.2019). https://juhaviitala.com/2016/12/20/mika-on-lohkoketju/.

[29] (2019), Proof-of-authority (luettu 15.5.2019). https://en.bitcoinwiki.org/wiki/Proof-of-Authority.

[30] Baliga A. (2017) Understanding blockchain consensus models. In: Persistent.

[31] (2018), Majority attack (luettu 9.5.2019). https://en.bitcoin.it/wiki/Majority_attack.

[32] Brown D. (2010), Standards for efficient cryptography 2 (sec 2) (luettu 13.5.2019). http://www.secg.org/sec2-v2.pdf.

[33] John M. (2016), Intel® software guard extensions part 7: Refine the enclave with proxy functions (luettu 15.5.2019. https://software.intel.com/en-us/articles/intel-software-guard-extensions-tutorial-part-7-refining-the-enclave.

[34] Hoekstra M. (2015), Intel® sgx for dummies (intel® sgx design objectives) (luettu 9.5.2019). https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx.

[35] Intel® software guard extensions (luettu 14.5.2019). https://software.intel.com/sgx-sdk.

[36] Optimized c library for ec operations on curve secp256k1 (luettu 14.5.2019. https://github.com/bitcoin-core/secp256k1.

[37] Library for the shamir secret sharing scheme (luettu 14.5.2019). https://github.com/dsprenkels/sss.

[38] Gordon J. (2018), Microsoft* azure confidential computing with intel sgx (luettu 17.5.2019. https://software.intel.com/en-us/blogs/2018/11/08/microsoft-azure-confidential-computing-with-intel-sgx.

[39] Aziz B. & Hamilton G. (2009) Detecting man-in-the-middle attacks by precise timing. In: 2009 Third International Conference on Emerging Security Information, Systems and Technologies, IEEE, pp. 81–86.

[40] Khandelwal S. (2019), First android clipboard hijacking crypto malware found on google play store (luettu 17.5.2019). https://thehackernews.com/2019/02/android-clickboard-hijacking.html.

[41] Barker E. & Dang Q. (2015) Nist special publication 800–57 part 3: Application-specific key management guidance. NIST Special Publication 800, p. 12.

[42] Brasser F., Müller U., Dmitrienko A., Kostiainen K., Capkun S. & Sadeghi A.R. (2017) Software grand exposure: SGX cache attacks are practical. In: 11th USENIX Workshop on Offensive Technologies (WOOT 17), USENIX Association, Vancouver, BC. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser.

[43] Chen G., Chen S., Xiao Y., Zhang Y., Lin Z. & Lai T.H. (2018) Sgxpectre attacks: Leaking enclave secrets via speculative execution. CoRR abs/1802.09085. URL: http://arxiv.org/abs/1802.09085.

[44] Wang W. (2019), Cryptocurrency hacks still growing — what does that mean for the industry? (luettu 15.5.2019). https://thehackernews.com/2019/05/bitcoin-ethereum-hacks.html.

[45] Christie C. & Dill E. (2016) Evaluating peers in cyberspace: The impact of anonymity. Computers in Human Behavior 55, pp. 292 – 299. URL: http://www.sciencedirect.com/science/article/pii/S0747563215301552.

[46] Feng J., Lazar J. & Preece J. (2004) Empathy and online interpersonal trust: A fragile relationship. Behaviour & Information Technology 23, pp. 97–106.